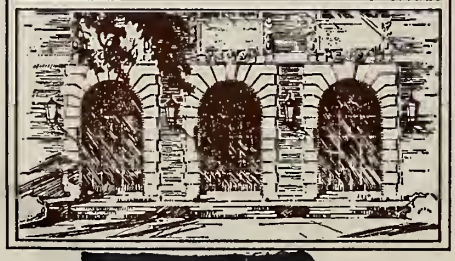


Math

UIUCDCS-R-74-633

DIGITAL RADIO AIDS FOR A FLIGHT SIMULATOR
RADIO MEMORIES - DISPLAY - CODE KEYER

by

Lucien Isidore Facchin

May, 1974

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-74-633

DIGITAL RADIO AIDS FOR A FLIGHT SIMULATOR

RADIO MEMORIES - DISPLAY - CODE KEYER

by

Lucien Isidore Facchin

Ingénieur E. S. I. E. E., Paris (France), 1972

May, 1974

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

This work was supported in part by the Institute of Aviation and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, May, 1974.

ACKNOWLEDGEMENTS

The author wishes to express his sincerest gratitude to his advisor, Professor W. J. Kubitz for his guidance, timely suggestions and continuing interest. The author also wishes to thank Mr. Lynn Staples, from the Institute of Aviation, who suggested the digital Radio Aids.

Further, the author wishes to thank Ms. Evelyn Huxhold, who typed the final version of the paper.

ABSTRACT

The research described here is part of a joint study between the Department of Computer Science and the Institute of Aviation of the University of Illinois. This study is a result of the need to consider alternatives to using general purpose digital computers for the solution of the entire aircraft simulation problem. The software costs of such systems have sky-rocketed in recent years while hardware costs continue to fall. Thus, serious consideration must be given to direct functional mechanization through the use of special purpose computers. The use of small dedicated computers may greatly simplify interfacing to the central control computer in addition to reducing the amount of software required by difusing the processing throughout the simulator. This document presents an implementation of some of the functions of such a system, the Radio Aids System.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

1. INTRODUCTION

1.1. Definition of the Basic Simulator

Since the thirties, when general aviation underwent a large expansion, it has been suggested that simulators be used in order to ease the flight instruction process, at least in the initial stage. One of the first trainers was the Link Trainer.

This trainer is a relatively simple device. It is composed of a series of pneumatic pumps and pushrods that can turn and bank a plywood box, whose inside is made to look like an aircraft cockpit. The pilot sits in the box and moves it around using appropriate stick or control wheel movements. The pilot learns the basic means for making the aircraft respond by using the control system, as well the rate with which an aircraft will respond to the pilot's commands. A trainer gives a pilot a good idea of how the instruments on the aircraft panel react to each movement of the control wheel. The pilot quickly learns how the speed and altitude of the aircraft are affected by different power settings and control wheel movements. The pilot learns which instrument to watch for precise monitoring of the behavior of the aircraft. With this knowledge, a pilot is better able to fly the first time he pilots an aircraft. In addition, there is the distinct advantage of being able to make mistakes in the Link Trainer during initial attempts without worrying about damaging the aircraft. In some cases, flight simulation is the only possible way to teach flying; for example, the landing on the moon or the training for air-to-air combat.

1.2. Present Systems

Since the first Link Trainer, aircraft have become more complex as have the simulators, which rely heavily on computers. Figure 1 shows a block diagram of a typical flight simulator system. It represents a cockpit which contains the aircraft instruments delivering analog signals. In order to be processed by a computer, these signals must be converted to digital form through an analog to digital interface (commonly called a linkage in the jargon of the flight simulator specialist). The data is then processed in a computer which contains data and programs; in fact, in several cases we need more than a single computer since aircraft have become more and more sophisticated. Associated with the computer is a console for the instructor who gives orders, decides changes of data in the simulated flight and checks the reactions of the trainee. The data coming from the computer is fed back to the cockpit through a digital to analog interface.

1.3. Problems Appearing in the Present Systems and a Possible Way to Decrease Some of Them.

Several problems have arisen in current flight simulators; two of them will be discussed here.

1.3.1. Software Cost

Flight simulators typically need several computers. Each time these computers change, the software must be rewritten, usually at great cost. During the past two decades, hardware and software have exchanged roles. At the beginning of the '60s, the hardware costs accounted for more than 50 per cent of the total computer system costs. Now the situation is reversing in favor of hardware, which currently accounts for less than 50

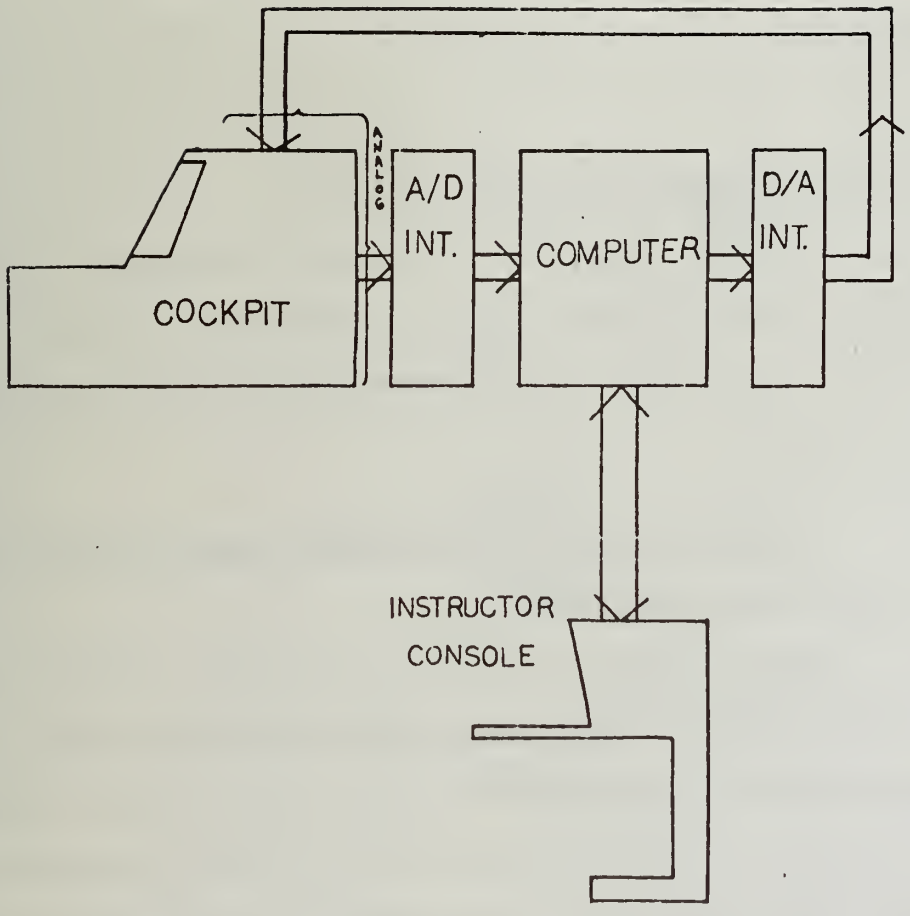


Figure 1. Block Diagram of a Typical Flight Simulator.

per cent of the total computer system costs. In future years, it seems that the share of costs due to hardware will continue to decrease. Due to the low cost of LSI hardware, more and more complex functions, usually performed by software, are being implemented in hardware form. Another argument for direct hardware implementation is reducing the time needed to perform the required computation.

1.3.2. Wiring from Cockpit to Interfaces

Due to the large number of airplane instruments, the amount of wiring connecting the cockpit to the interfaces is very large. In addition, the distance between them may reach several tens of yards making the cost very high.

1.3.3. Proposed Solution to These Problems (Special Purpose Computer or Direct Functional Mechanization)

Among the other functions to be simulated, the Radio Aids, which are the main navigation instruments, may be implemented without going through a classical computer but by using a special purpose computer as shown in Figure 2. The data coming from the radio instruments are directly coded in binary form, thus eliminating any analog to digital and digital to analog interfaces. This is made possible by using an algorithm called CORDIC which implements trigonometric functions (sin, cos, arctg,... log) using only a small amount of hardware. The idea of CORDIC was originally proposed for realizing the transformation $(X, Y) \overset{\rightarrow}{\leftarrow} (R, \theta)$ to compute the range of the plane. The use of Read Only Memories allows one to implement several functions using the same basic elements (adders, shift registers) without duplication, by writing new codes in those ROMs.

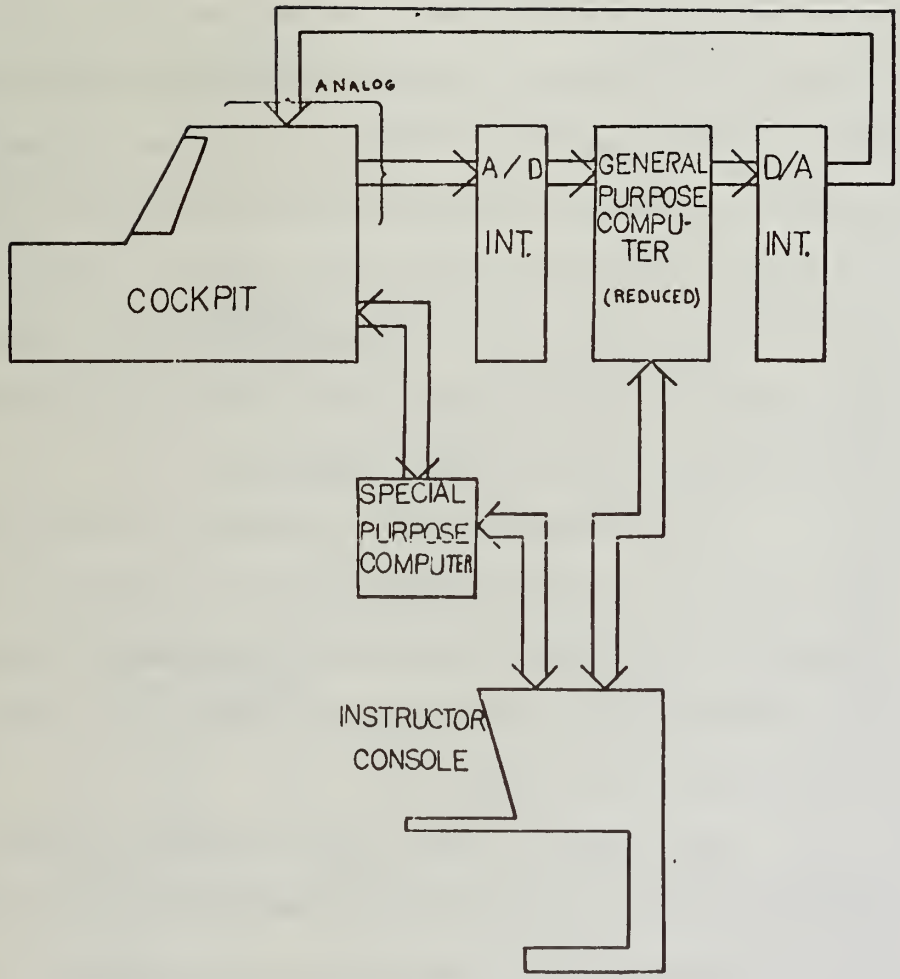


Figure 2. System Under Study.

1.3.4. The Radio Aids and a System Using CORDIC

Radio navigation is now the most common form of navigation. In addition to weather information, radios help the pilot to determine his position from specific radio stations. In the cockpit, the pilot has available a set of radios which are used for different purposes. One class of radios are those used for the airway system and are themselves divided into two groups: the VOR (Visual Omni-Range) which informs the pilot on which radial he is flying, and the Low Frequency Radio Beacon (Distance Measurement Equipment) which determines the distance by computing the time required for a pulse of radio energy to travel from the plane to a ground station and return. For landing, the radio instruments used are the ILS (Instrument Landing System) which provide the information a pilot needs for landing in poor visibility conditions.

A very simplified block diagram of the radio aids simulator is given in Figure 3. The system monitors the radios in the cockpit. At the request of the pilot, the Radio Aids simulator must provide to the cockpit the information that the pilot would receive if he were in a real flight (such as Morse code, range, ...).

The system has a set of Radio Memories (PROMs) which can be programmed and which contain the data characterizing the stations within range of the aircraft. These memories are scanned so as to find all stations receivable by the radios. The Cordic Unit uses data from the memories in conjunction with the aircraft position coordinates to perform several calculations such as the range computation. A particular function of the Radio Aids is the Code Keyer which must provide the identifying code for each radio station which is received.

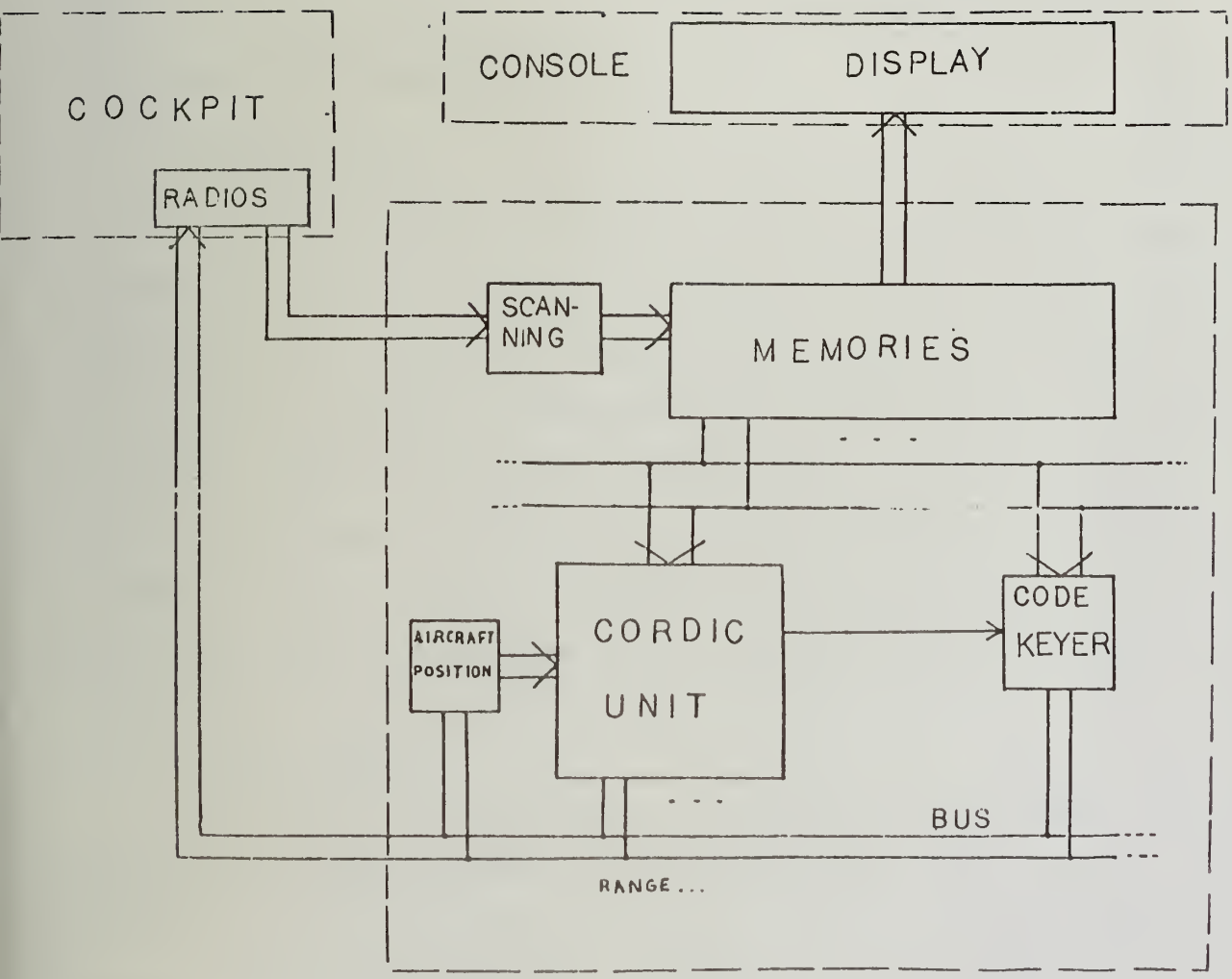


Figure 3. Block Diagram of the Radio Aids.

1.3.5. Radio Memories, Display and Code Keyer

The following pages present solutions which have been proposed and the implementation of one of them for the following functions:

a) Radio Memories

b) Display

c) Code Keyer.

2. THE RADIO MEMORIES

Introduction

In the introduction, a system using the Cordic algorithm was discussed in its general configuration. One of its functions will be considered here and a design is proposed for the Radio Memories system. This is described in the following pages.

2.1. Radio Memories

A flight simulator is a completely self-contained device, having no communication with its environment. Therefore, we must store the radio stations parameters in some way. By radio station parameters, we mean: its X, Y coordinates (latitude and longitude), its altitude, the type of station, the magnetic variation, the power of emission, and the morse code letters which there are three. In this system we can differentiate two functions: the PROMs which are the memories themselves, and the scanning device which selects one address of the memories in order to load or check its contents.

2.2. Memory Elements

The first factor to consider for the memories is the number of words they should contain (a word corresponds to one station). This is determined by the autonomy of the plane used. In the case of small planes, a radius of 300 miles has been considered adequate. An investigation of the number of radio stations within a 300 mile radius has been done and it indicates a number on the order of 200. Therefore a possible configuration is to use 256 word memories (e.g. INTEL 1702A). The data to be stored are:

X: the first coordinate of the station with respect to a given origin; X is either positive or negative, coded in two's complement.

Y: second corrdinate of the radio station. It has exactly the same format as X.

Z: altitude in feet and essentially a positive number,

magnetic variation which indicates the difference between the geographic and magnetic longitudes.

radiated power

type (VOR, radar, DME, ...)

Morse code letters which are three five bit words.

X and Y are the variables which require the largest number of bits (24) and thus require 3 PROMs. Therefore, the use of a 24 line bus has been considered. For the altitude Z (16 bits) and the magnetic variation (8 bits), the data is spread from the first to the 24th lines. A set of control lines provide signals which enable the PROMs containing the desired data. See Figure 4.

2.3. Scanning Device

A radio is an element which has two aspects (see Figure 5). One is the tuning system. When a frequency is selected a binary number is generated and placed on the bus in order to select corresponding data in the PROMs. The other aspect of a radio is the visual display which differs with the type of radio. This visual information is obtained after processing of the data contained in the PROMs for that radio.

The analysis of the scanning device has progressed through several stages and in the following pages, several approaches will be presented.

2.3.1. A First Approach

A first approach considers a limited number of radios (up to 8) and supposes that each frequency corresponds to only one radio station.

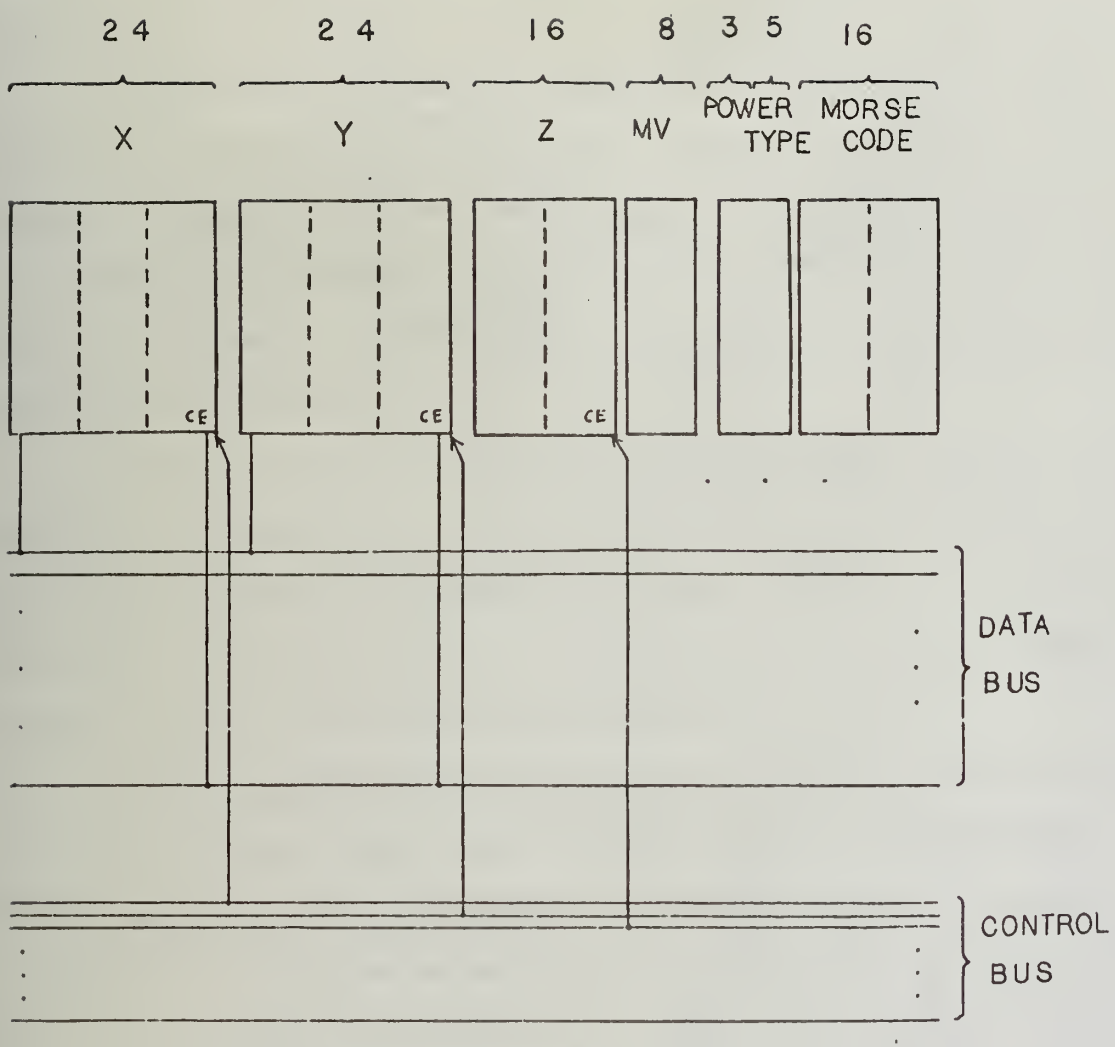


Figure 4. Memories Configuration.

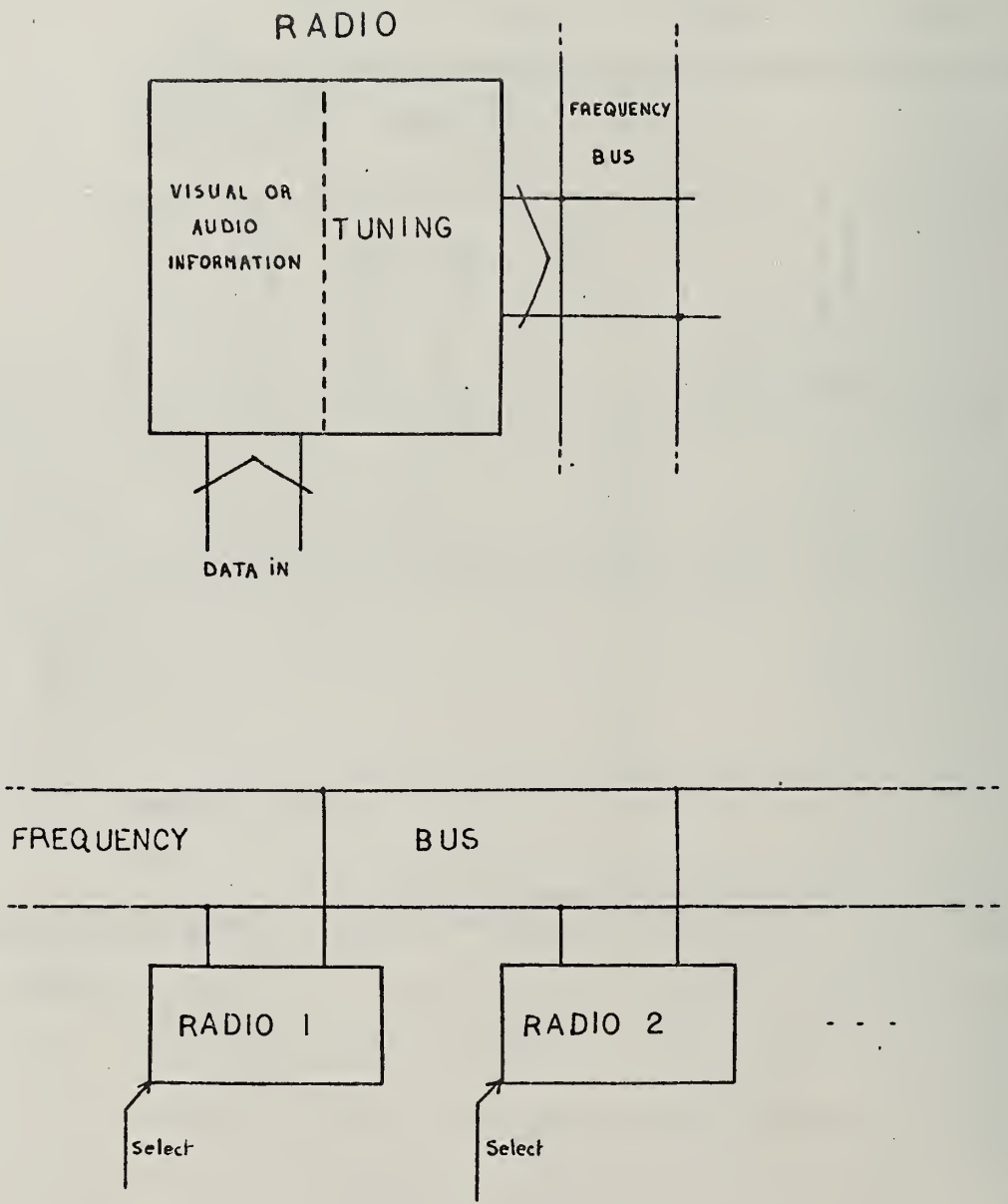


Figure 5. Functional Diagrams of Radios.

In addition we do not consider the interaction with the computing element (CORDIC unit). Each frequency may be represented by 8 bits. Refer to Figure 6.

We have a set of radios each delivering an 8 bit number. Each radio must be considered individually as it addresses the memory. This is implemented by using eight 8 bit multiplexers. Each multiplexer selects one bit among 8 which belong to all radios. A clock drives a 3 bit binary counter which sequences the multiplexers selecting the radio frequencies individually and consecutively.

This is a very simple approach to the system which must be modified for the following reason. Since the number of radio stations is much greater than the number of frequencies available, there exist frequencies used by more than one radio station. Since we are addressing the PROMs by the radio frequency, the preceeding system must be modified.

Figure 7 indicates, in an X, Y plane, the position of the aircraft and two stations noted 1 and 2 which are using the same frequency f_A. When a frequency is set on one of the radios, we are to determine which of the stations is the one we are receiving (in simulation). In addition, we suppose that the plane cannot receive the signals emitted by more than one station having the same frequency. The selection of the right station is realized as indicated. First, the distance d1 from the plane to the first station found in the PROM (let it be station 1) for the given frequency is determined. Then, distance d2 from the plane to the next station with same frequency (let station 2) is computed. If they are the only stations using that frequency, we compare d1 and d2 and the station having the smallest distance is the right one (for $d1 < d2$ it is station 1).

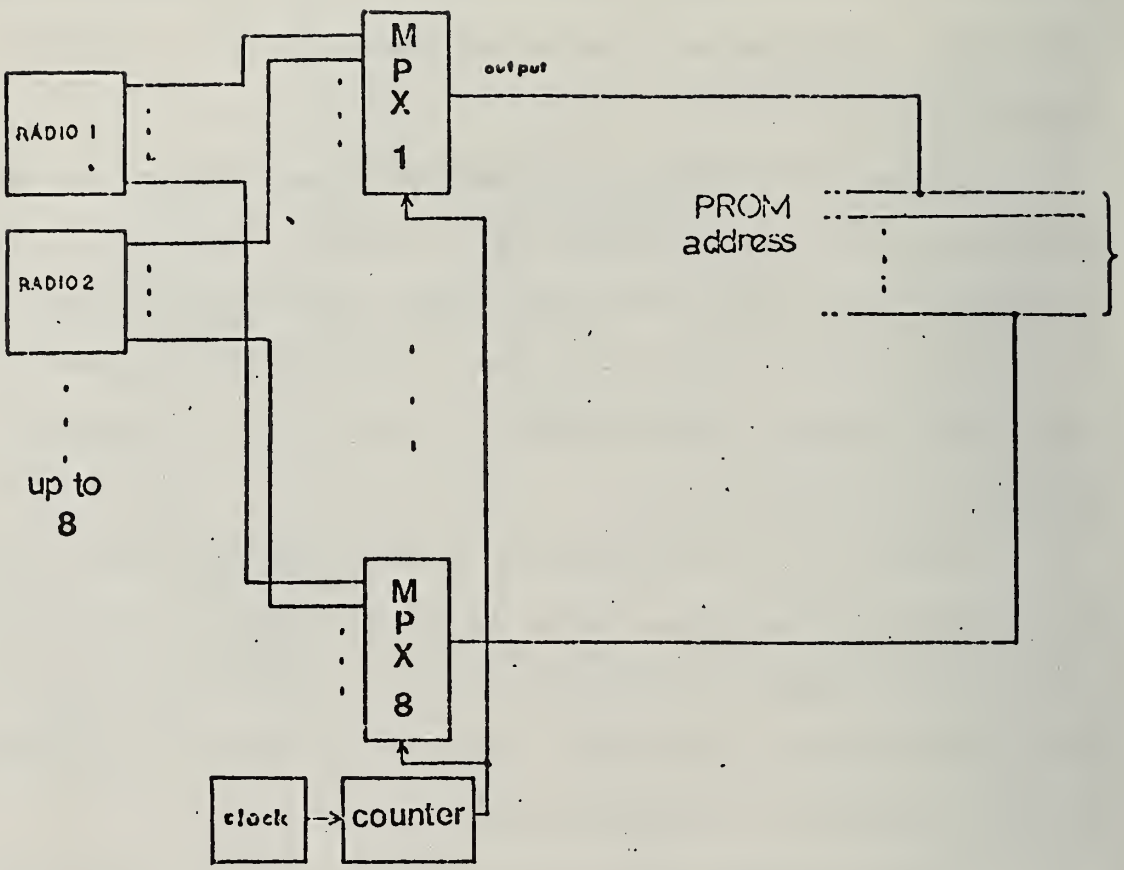


Figure 6. First Approach of the Scanning.

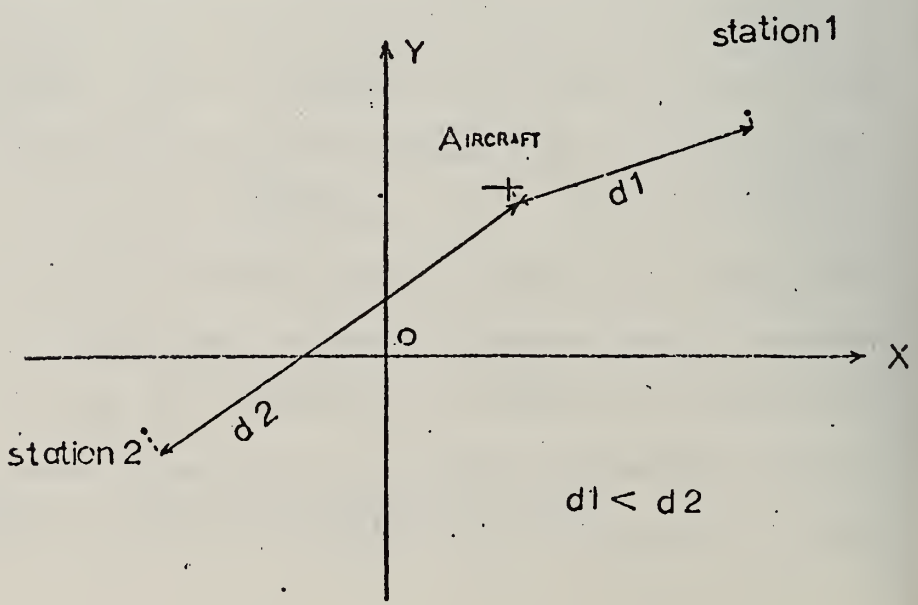


Figure 7. Case of Stations Using the Same Frequency.

2.3.2. A Second Approach

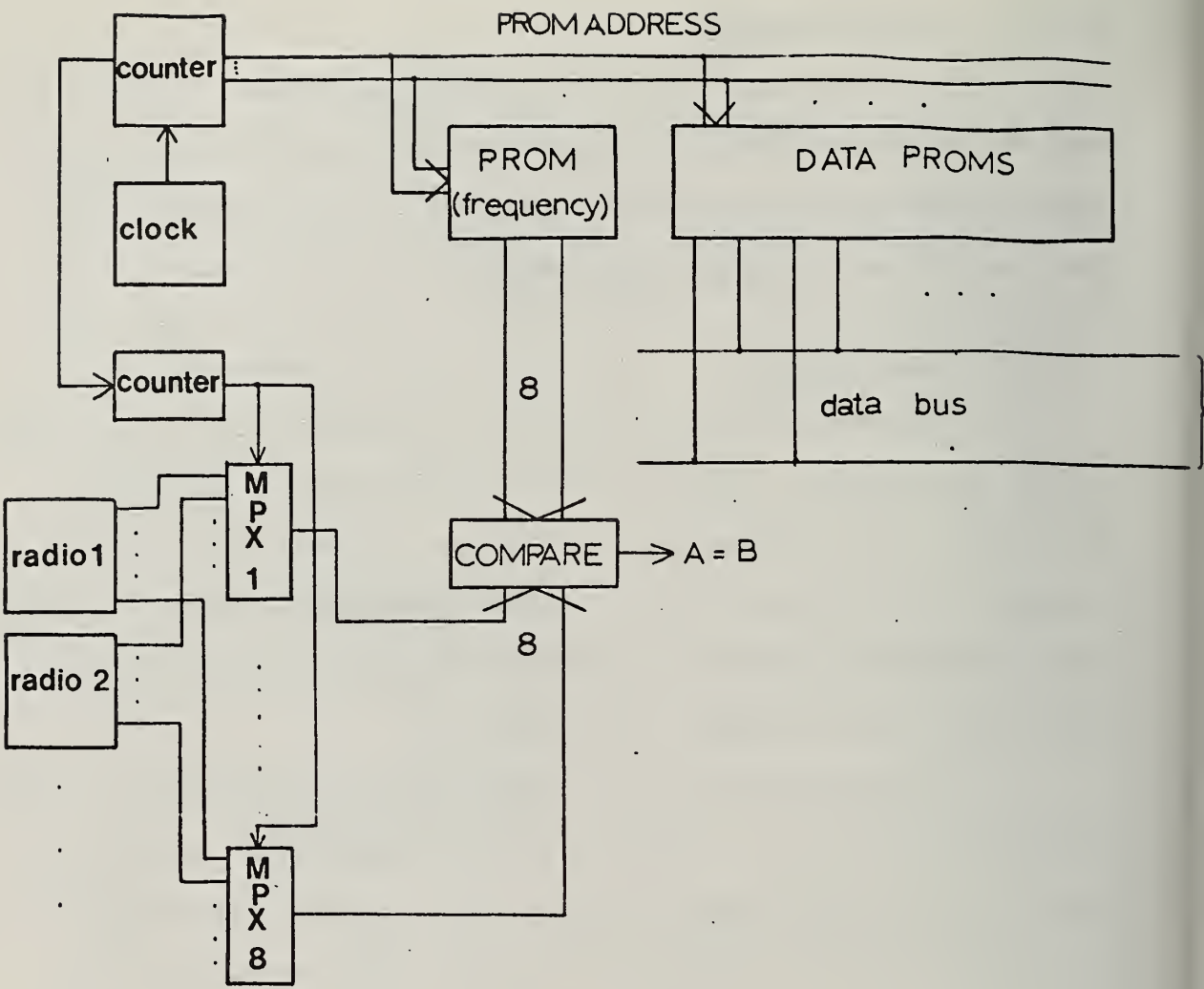
At this point, we will use the above constraint. We still suppose that there are only 8 bit frequencies, and as before we do not consider the interaction with the computing unit. Several solutions have been proposed which will be roughly sketched.

2.3.2.1. First Solution

This solution uses one additional ROM which contains the frequencies of the stations. Figure 8b shows this PROM which contains the frequencies. For a frequency f1, we have two stations and therefore the frequency f1 will appear twice in the PROM (addresses 8 and 91 for example). When the PROMs are selected at these addresses, the data corresponding to these stations are available on the data bus.

The system works in the following manner: one radio is selected using multiplexers and its frequency enters a comparator. During this time, an 8 bit counter scans the entire radio memory in order to test the frequency which is contained in each word. This frequency is the second input of the comparator. When both frequencies are identical, the comparator generates a signal A = B indicating that we have found a station having the same frequency as the one corresponding to the radio considered. At this time we can use the data presently on the data bus. The counter scans all the addresses. When the last address has been selected, we again start the cycle and a signal is generated in order to select the next radio (this is implemented by incrementing a 3 bit counter which addresses the multiplexers). The same procedure as indicated above is repeated for the new radio and so on. See Figure 8a.

An analysis of this system shows a great advantage to this method of scanning: any station data (frequency and parameters) can be placed



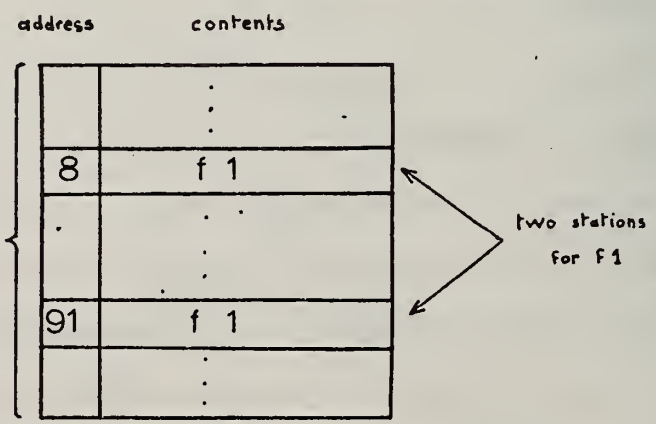


Figure 8. First Solution.

at any location in the PROMs, which means great ease of maintenance. When new radio stations are constructed, the parameters of these are loaded into the PROMs at the next location without any removal or rearrangement of the data already inside. However, the system has an important drawback in this form: for each radio, we have to scan the whole 256 words among which one or two or occasionally more exist. In the next approach, a solution is proposed in order to avoid this drawback.

2.3.2.2. Second Solution

Another solution (with two forms) is now presented which attempts to provide faster scanning of the stations in which we are interested.

Compared to the preceding one, it does not use a PROM for storing the frequencies and therefore implies that the parameters of the radio stations are ordered in a particular way.

Two schemes of ordering the data are presented: the first one provides automated scanning while the second is semi-asynchronous.

First Scheme

This scheme assumes 8 bit frequencies and that, for a given frequency, there exists no more than four stations. Presently this assumption proves to be close to the facts if we consider an area of 300 miles radius; using Champaign as center it was found that only one frequency was used by more than four stations. An additional supposition is that no more than 256 stations exist in the zone. A block diagram of the system is shown in Figure 9.

The 8 bit radio frequency is divided into two parts. The first one, consisting of the 6 most significant bits, is used to address the 6 most significant bits of the "address PROMs" (to be defined later). The second part (two least significant bits) is decoded through a 2 to 4 line

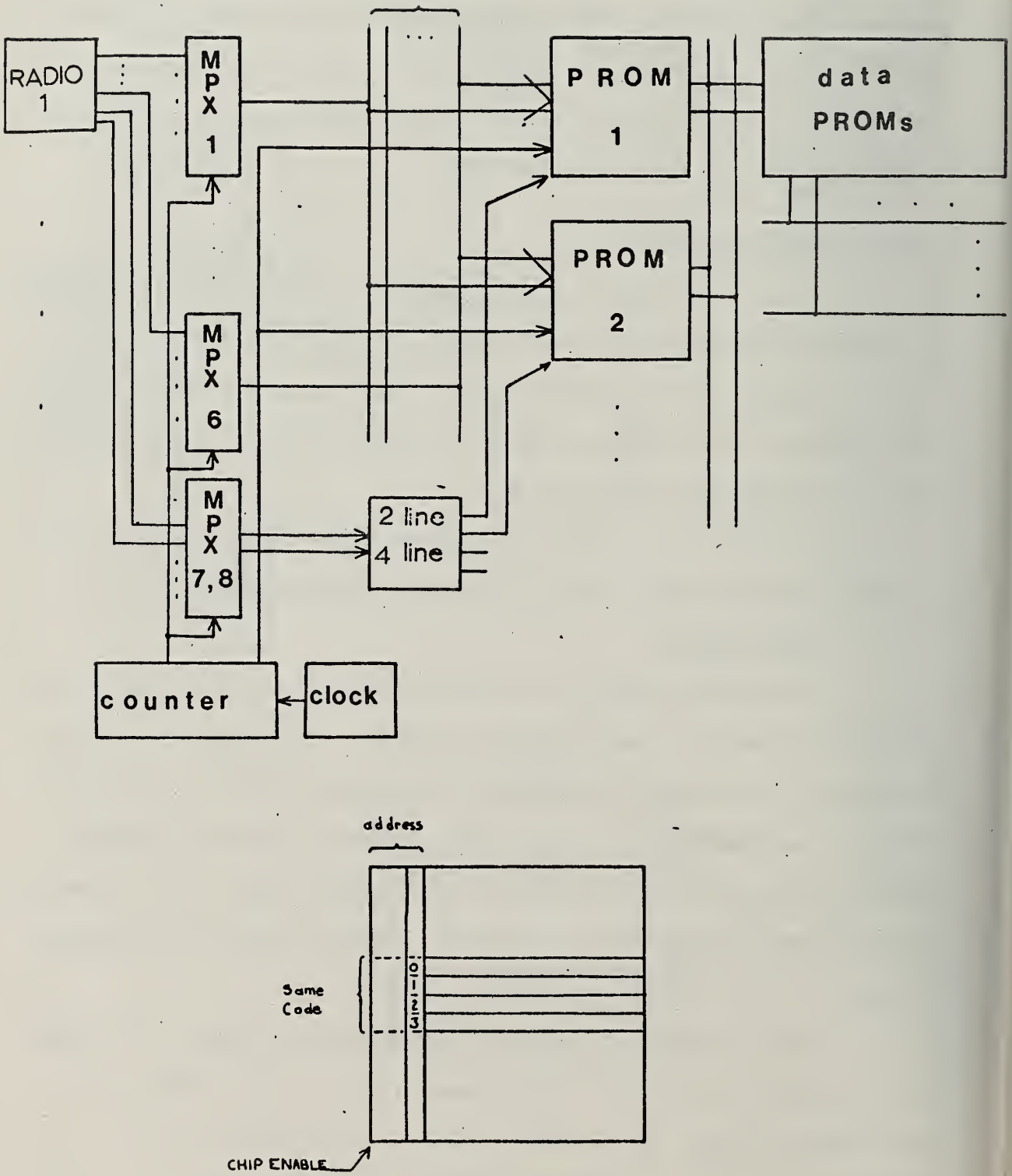


Figure 9. Second Solution (First Scheme).

decoder. Each of the four lines is used to enable one of the four address PROMs. Now, let us define the address PROMs. They are up to 4 in number and each of them is selected as indicated above. For addressing, the two least significant bits are generated by the two least significant bits of a 5 bit counter. The three most significant bits of this counter are used to select one of the radios. Consider the case where the counter begins to select one radio. Its corresponding 8 bit frequency is therefore selected by the mean of multiplexers. Bits 7 and 8 determine the address PROM of that frequency. Meanwhile, the two least significant bits of the counter vary from 00 to 11, therefore providing an automatic scanning of four consecutive words of the enabled address PROM. Each address PROM is loaded with 8 bit words which are the addresses of the PROMs containing the parameters of the radio staions. The words are organized in groups of four, each group containing the addresses for stations using a given frequency. If only one station is using a frequency, three out of four words will be loaded with dummy data corresponding to a dummy address of the data PROMs.

Let us analyze the efficiency of such a system. Its main advantages are automation and rapidity. If we consider 8 radios, these are completely scanned in 8 x 4 = 32 time units compared to the preceding system which requires 256 x 8 = 2048 time units. However, the system seems more complex since it requires more integrated circuits. One drawback is the inefficient use of the address PROMs. We have decided to assign four words for each frequency but the average number of stations using one frequency is between two and three. This means that only 60% of the words can be used. Assigning three words for each frequency would increase this efficiency, but the system would require more hardware since we are using binary representation. Another drawback is the fact that we have assigned a fixed number, four, for the maximum possible number of stations

using a given frequency. This is the main limitation of the system. However, it has the advantage of simple maintenance since, when we have to load data for new stations, the data already inside is not removed or rearranged.

Second Scheme

Another scheme, which is as fast as the preceding one, but which avoids the limitation on the maximum number of stations using one frequency is now presented. Refer to Figure 10.

We suppose, in addition, that we may need up to four address PROMs. Therefore we choose a 10 bit frequency. Eight of them are used to address directly the address PROMs; the other two, using one 2 to 4 line decoder, provide an individual selection of these PROMs. In addition, the 8 bits provide a code which points to the word containing the first station using a given frequency. When a radio is selected by a set of multiplexers, the 8 bit frequency presets an 8 bit counter addressing the "address PROMs" and at the same time enables the clock which is monitoring this counter. The outputs of the address PROMs are fed back into a device which detects a dummy variable used to indicate that there exist no more stations at that frequency. When such a dummy code is detected, the 8 bit counter is stopped while enabling the 3 bit counter which selects one radio among the eight.

This method has several advantages: it is very fast (faster than the preceding ones); it has good storage efficiency which improves as the number of stations using the same frequency increases (we loose one word for each frequency). However several drawbacks can be pointed out. The first one is the fact that the system is not very automated, needing more hardware and controls. But the major problem with this scheme is its

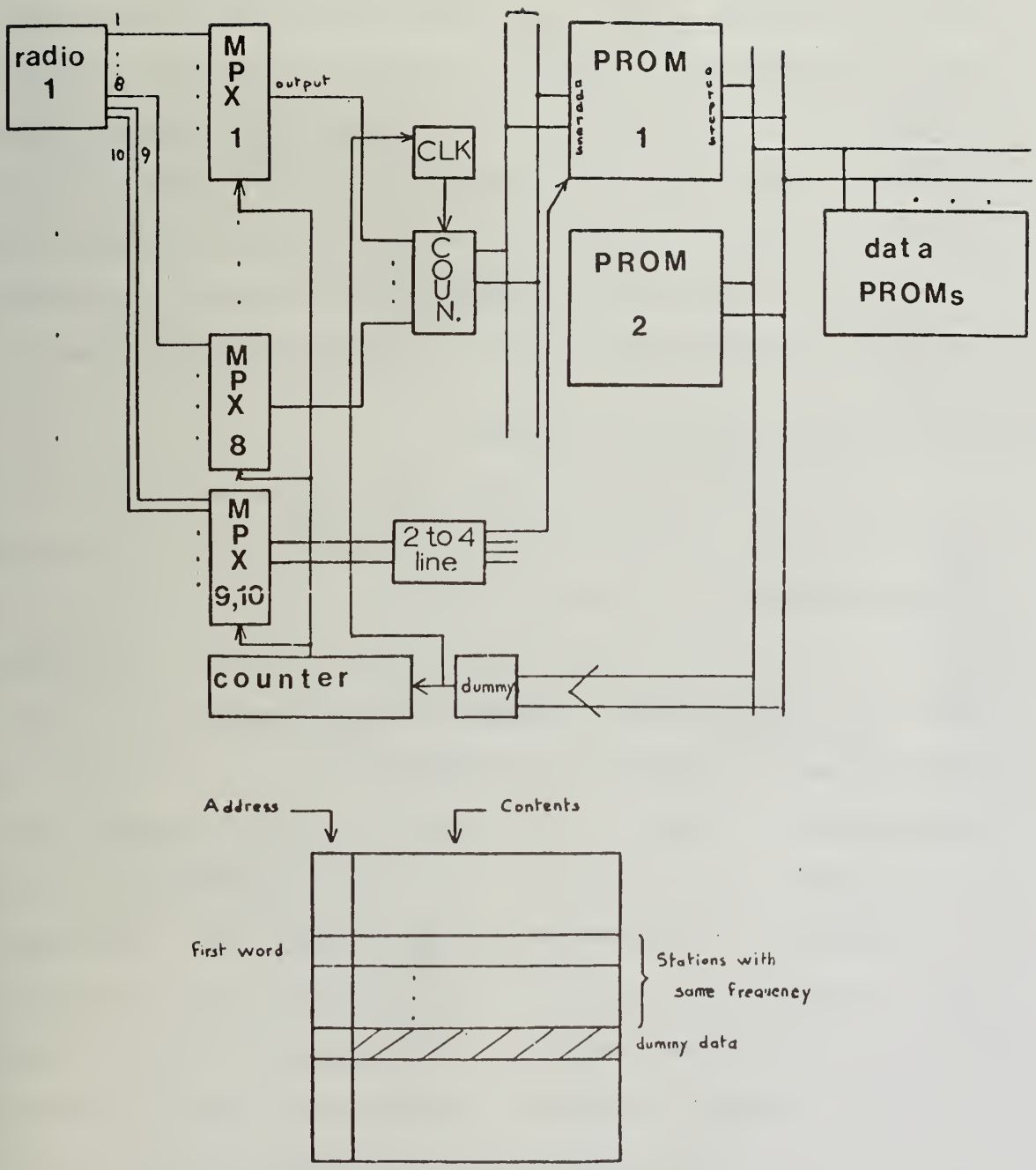


Figure 10. Second Solution (Second Scheme).

impractical maintenance. To load a new station at a frequency which is already used, we must load the address in place of the dummy word; this dummy word is written in the next location and the rest of the contents must be shifted down in the address memory, which requires substantial effort. This rearrangement of the contents may be avoided by using the following procedure. Instead of having a dummy code we load the address to which the 8 bit counter has to jump in order to find the next set of stations using the same frequency. This avoids removal of data everytime new stations are loaded in, but requires more control logic, which makes the system even more complex since it is already not very automated.

2.3.3. A Third Approach (Figure 11)

In the first approach, a very simple system was proposed. In the second approach, we took into account the fact that more than one station could use the same frequency. Now, we consider a system which is one more step toward a final design. The system uses the first solution indicated in the second approach (the method using a comparator). It was noted that the main drawback of this method was its long cycle time (while possessing ease of revision). A method which avoids this problem, while using the same principle, consists of scanning the entire memory (256 x 8) to determine all the addresses required by the CORDIC unit. These addresses are loaded into an 8 word RAM one after another until completion. When all radios have been processed, the RAM is cycled, each time providing a station address for each of the radios tuned. When a new radio is tuned (or the frequency changed), we set a flag which indicates that we want to repeat the scanning in order to refresh the addresses contained in the RAM. Let us analyze the operations involved in an entire cycle. When the flag indicating that the operator has selected a radio, the first radio

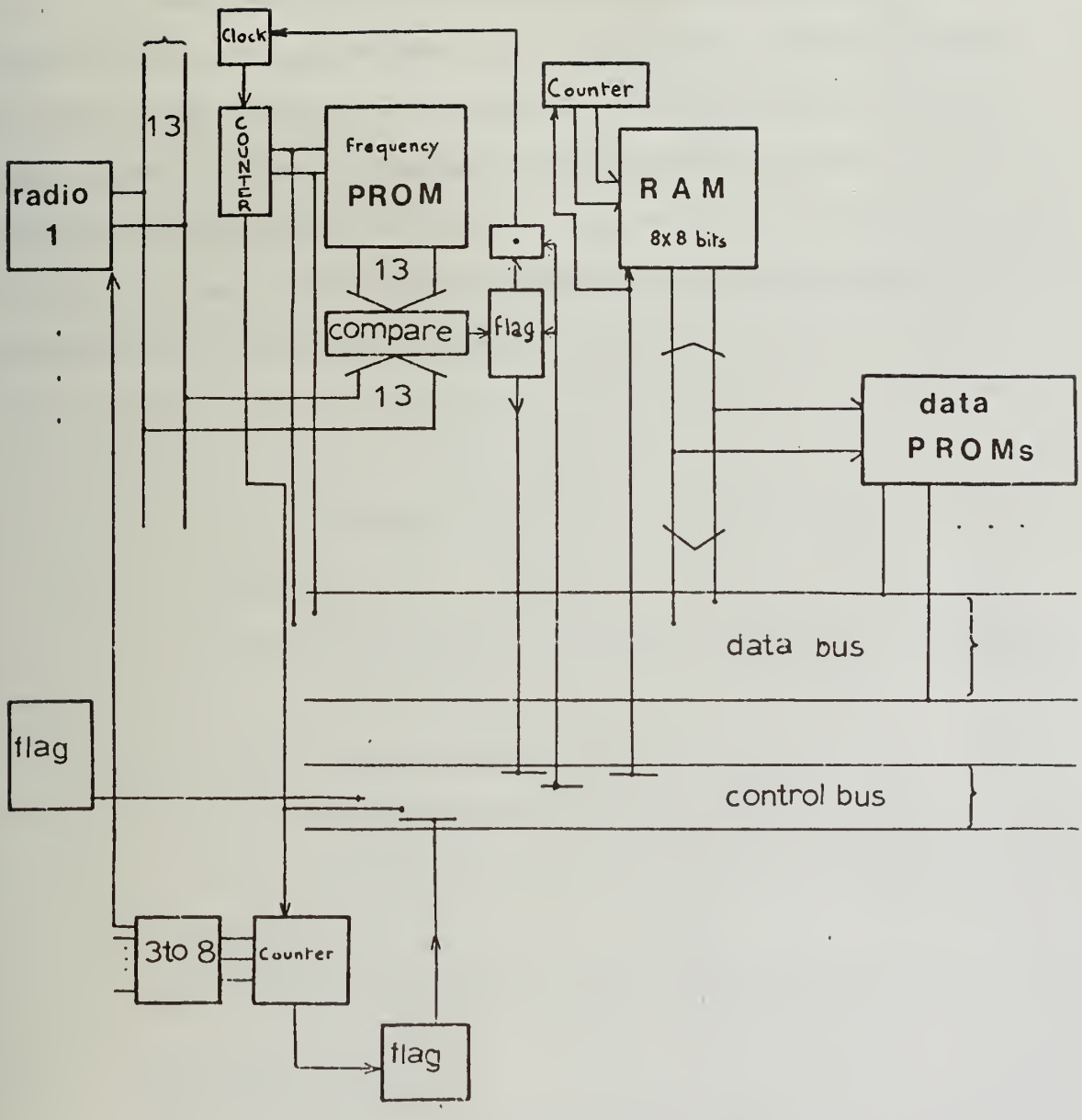


Figure 11. A Third Approach.

is selected and scanning of the frequency PROMs begins. When the comparator indicates equality, a flag is set which stops the scanning and activates a control line which causes the address to be saved and signals the CORDIC unit that it must compute the plane to station distance. When finished, a control line clears the flag and the scanning proceeds and so forth. When one cycle has been completed, CORDIC determines the right station (smallest distance) and loads its address into the RAM. When finished, the frequency PROMs are scanned again to determine the next station and so forth until all radios have been taken into account; at that time, we are ready to use the RAM for the cycling which is now of only 8 words.

3. THE DISPLAY

A single display will be used for the entire system. It is capable of displaying any of the data coming from the 24 bit data bus. The data to be displayed is determined by the operator by means of a selector. This selector provides lines which feed the control bus as indicated in Figure 12.

The data on the bus is in two's complement representation. In order to drive the display, it must be converted to BCD, with decimal point and sign.

Let us define the types of data to be displayed:

- data corresponding to a positive binary number, without any decimal point
- distances, either positive or negative, involving a decimal point
- distances to be converted in feet
- data to be read in angles (37768_{10} = 180.00 degrees)
- the three letters of the Morse code which must appear simultaneously.

Thus there are five types of data to be converted to BCD.

3.1. Conversion to BCD

The main problem to be solved is the conversion of a binary number, coded in two's complement when negative, into BCD. Several possibilities are offered and others have been considered.

3.1.1. Conversion Using Counting (Figure 13)

This is the simplest conversion. It uses two sets of counters; The first one is a set of binary counters; the second one uses BCD counters.

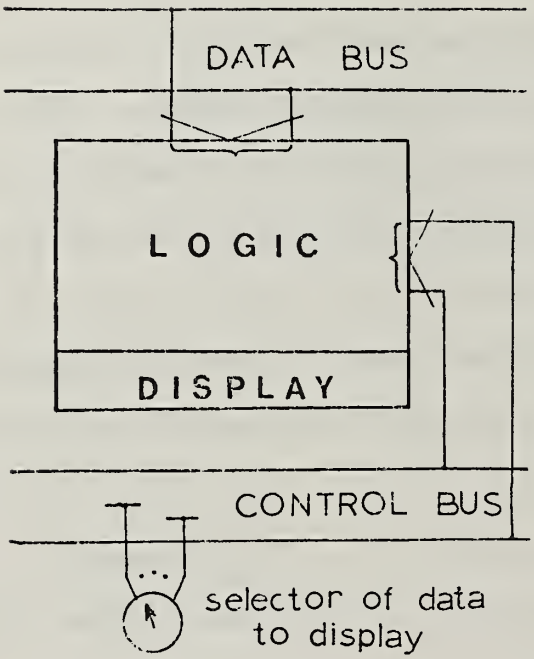


Figure 12. Block Diagram of the Display.

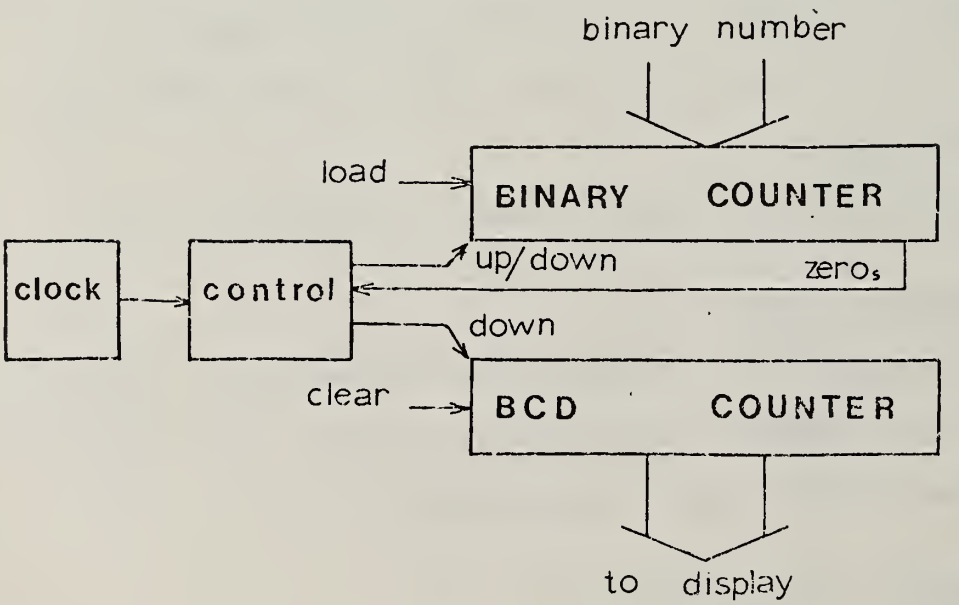


Figure 13. Conversion to BCD Using Counting (Basic Operation).

The conversion procedure works in the following manner: the number to be decoded is loaded into the binary counter while the BCD counter is cleared. Then a clock common to both counters is enabled causing the binary counter to count down and the BCD counter to count up. When all zeros are detected in the binary counter, we stop counting and the number coded in BCD appears at the outputs of the BCD counter.

The same hardware may also be used to convert a negative number (two's complement) into BCD; in this case, the binary counter counts up instead of down and stops counting when all zeros are detected.

This system has the advantage of being available for other kinds of operations. For example, as noted, some of the data corresponding to distances expressed in miles may be converted into feet. In the PROMs, the data is stored in binary and corresponds to miles x 100 (e.g. $10010000l_2$ = 2.89 miles). If we desire to display the distances in feet (a mile is 6000 feet in aviation), we must either multiply by 6000 or by .6 and move the decimal point four places to the right. The factor .6 is chosen so that we are able to use a binary rate multiplier (SN 7497) which may be cascaded to obtain better accuracy. Therefore, in order to perform the previous transformation in feet, the BCD decoder will be provided with a rate multiplier which modifies the clock frequency which operates the BCD counter. This type of converter may be used to display degrees, which also involves a rate multiplier. See Figure 14.

This system, although using a small number of chips, has the great disadvantage of being slow, especially if we desire to convert 24 bits.

3.1.2. Conversion Using Combinatorial Logic

In order to use a converter having no clock and thus very fast conversion speed (determined by the propagation rate through TTL logic),

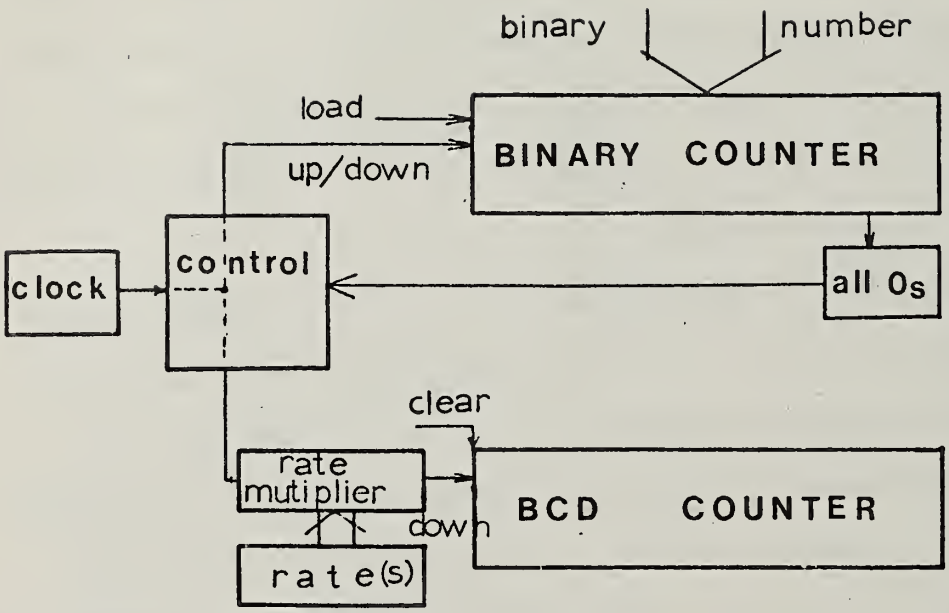


Figure 14. Conversion to BCD Using Counting (Extended Operations).

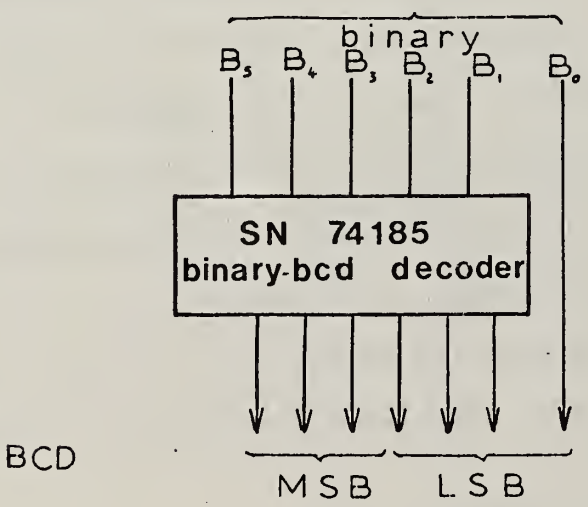


Figure 15. Conversion Using Combinatorial Logic.

the add/subtract-three algorithm implemented with ROMs (SN 74185) may be used. However, the number of packages required grows rapidly with the number of bits: 4 for 9 bits, 8 for 12 bits, 16 for 16 bits, and for 24 bits, the number of packages becomes prohibitive. Figure 15 shows one SN 74185 ROM which can convert up to 6 bit binary numbers.

3.1.3. Conversion Using BCD Adders

Another possible implementation of binary to BCD conversion using combinatorial logic is the following: use of ROMs and BCD adders. Figure 16 indicates the implementation for a 16 bit number. The first 8 bits provide the numbers from 0 to 255, thus requiring 2 ROMs each for the units, tens and hundreds. The next 8 bits address 3 ROMs since we need units to ten thousands. Five BCD adders are required to perform the additions of the units, tens and hundreds provided by the two sets of ROMs. This system can be extended to 24 bits but it requires a large number of ROMs and BCD adders. In addition it is not flexible.

3.1.4. Conversion Using ROMs and a Counter (Figures 17 and 18)

This scheme has some similarities with the preceding one, except that it requires more time and can be used for many kinds of transformations. Its principle is the following: the binary number is loaded into a shift register. A set of ROMs (4 for 24 bits) contains the BCD code for 2^{23}, 2^{22}, ... 2^{1}, 2^{0}. The clock shifts the shift register right and, at the same time, decrements a counter initially set at 24. For each digit, we use a BCD adder whose inputs come from the ROMs, and whose outputs are one input of a 2 input multiplexer with storage. The second input of this device is 0. The output of the serial output shift register selects either 0 or the data coming from the BCD adder. When the counter reaches 0, the clock is inhibited, and the operation is complete. The system has the

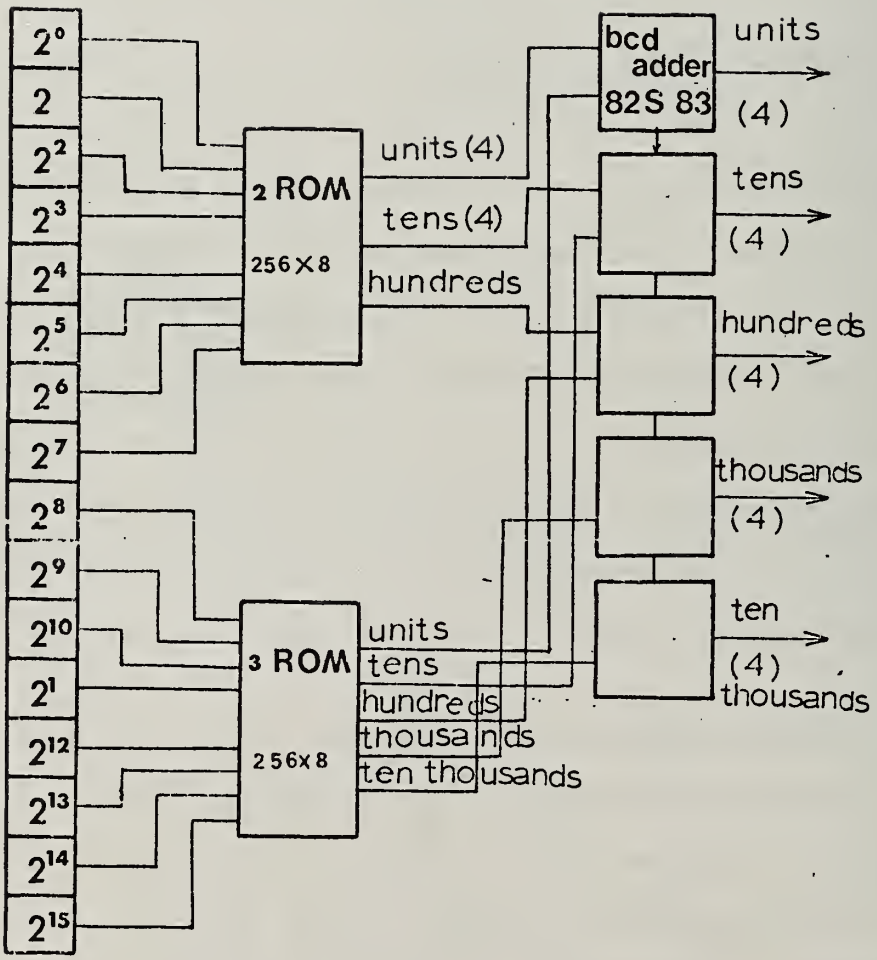


Figure 16. Conversion Using BCD Adders.

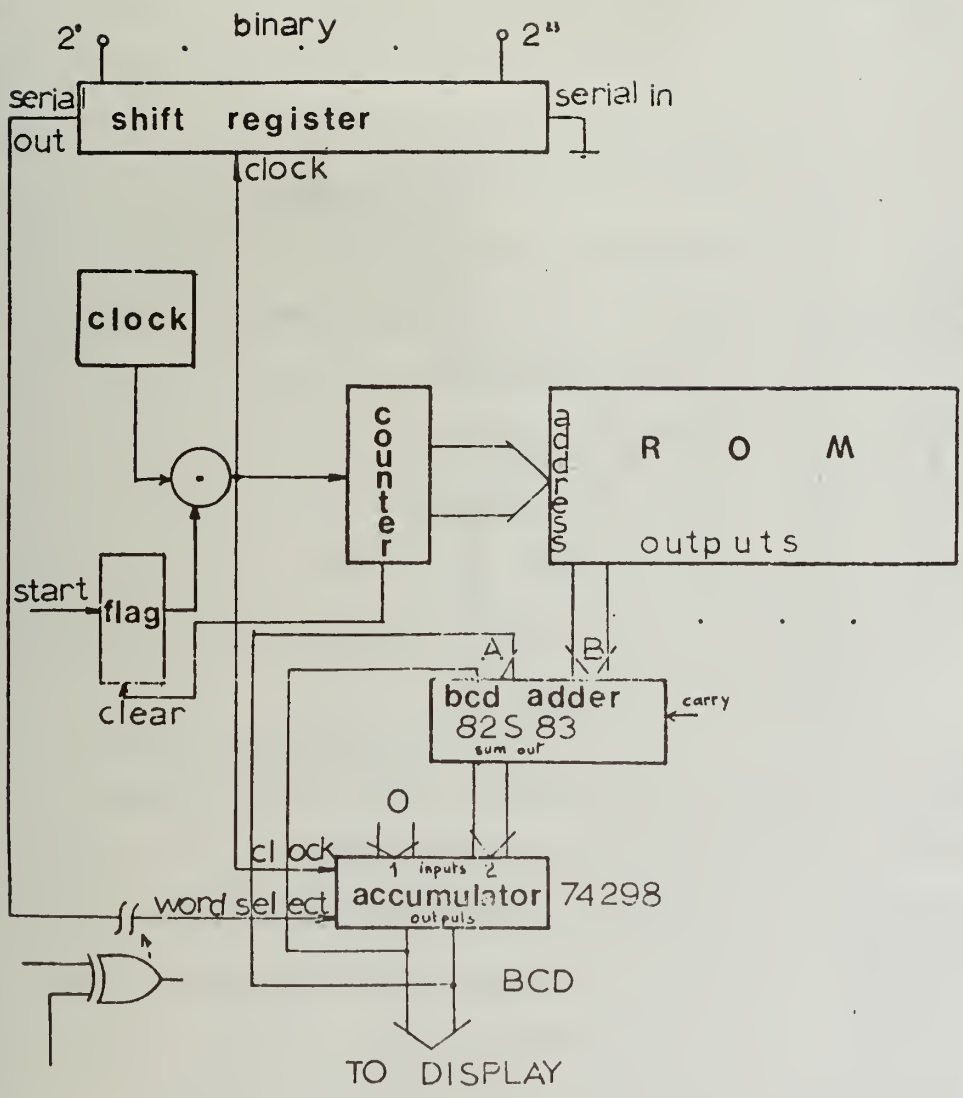


Figure 17. Conversion Using ROMs and a Counter.

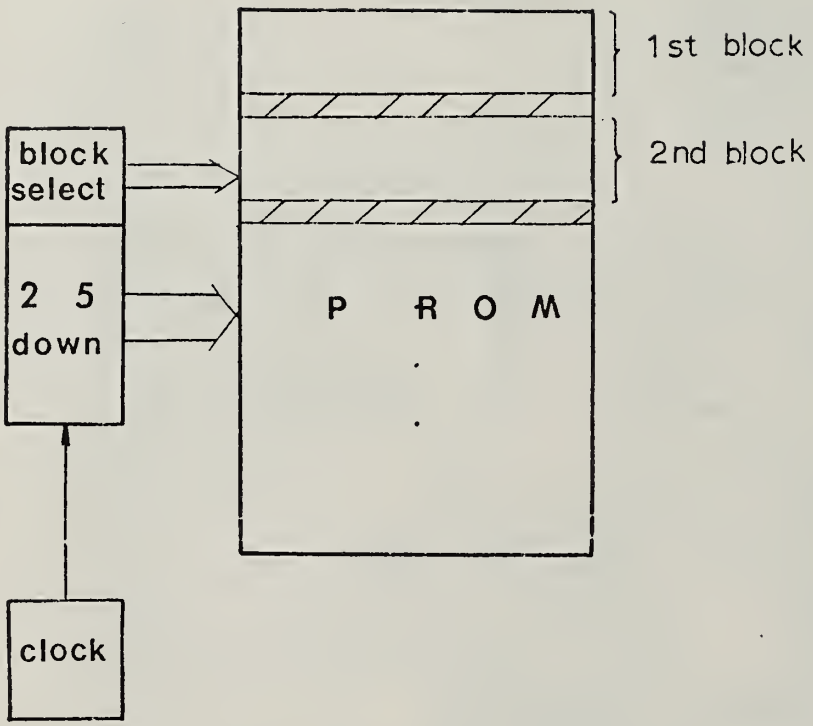


Figure 18. Converter Used for Multipurpose Conversions.

great advantage of being useful for binary numbers which are negative and coded in 2's complement. For this case, a 1 appears in the 24th bit. Figure 17 indicates an Exclusive OR circuit whose inputs are the outputs of a latch containing the 1 of the 24th bit and the current bit out of the shift register. In the ROMs, we use 25 words instead of 24 and the last one contains a one. When the 25th word of the ROMs is addressed, if the number is positive, 0 is added to the accumulator; if negative, the 25th word is added. Another interesting aspect of such a circuit, is the fact that it can be used for the conversion to feet or the angular representation as indicated above. The angular representation is as follows: 180° = -180° which corresponds in binary to 100...0; 90° is obtained by shifting right: 010...0 and so on. Therefore when 100...0 appears, we have to display 180.00 (using two decimal digits for greater accuracy).

3.2. Solution Implemented

Figure 19 shows a block diagram of the solution which has been implemented. It is a hybrid system, consisting of a set of binary and BCD counters for most of the conversions and of three 74 185 ROMs when it is desired to simultaneously display the three Morse code letters (the ROMs are used in order to avoid additional logic).

The system consists of two parts, the conversion circuits and the control logic.

3.2.1. The Conversion Circuits (Figure 20)

The circuits for the Morse code letters consist of a set of latches: 2 hex latches (SN 74174) and one quadruple latch (SN 74 175). They store the data coming from the data bus. The outputs of the latches are in groups of five and energize the 74 185 ROMs used for the binary to BCD conversion.

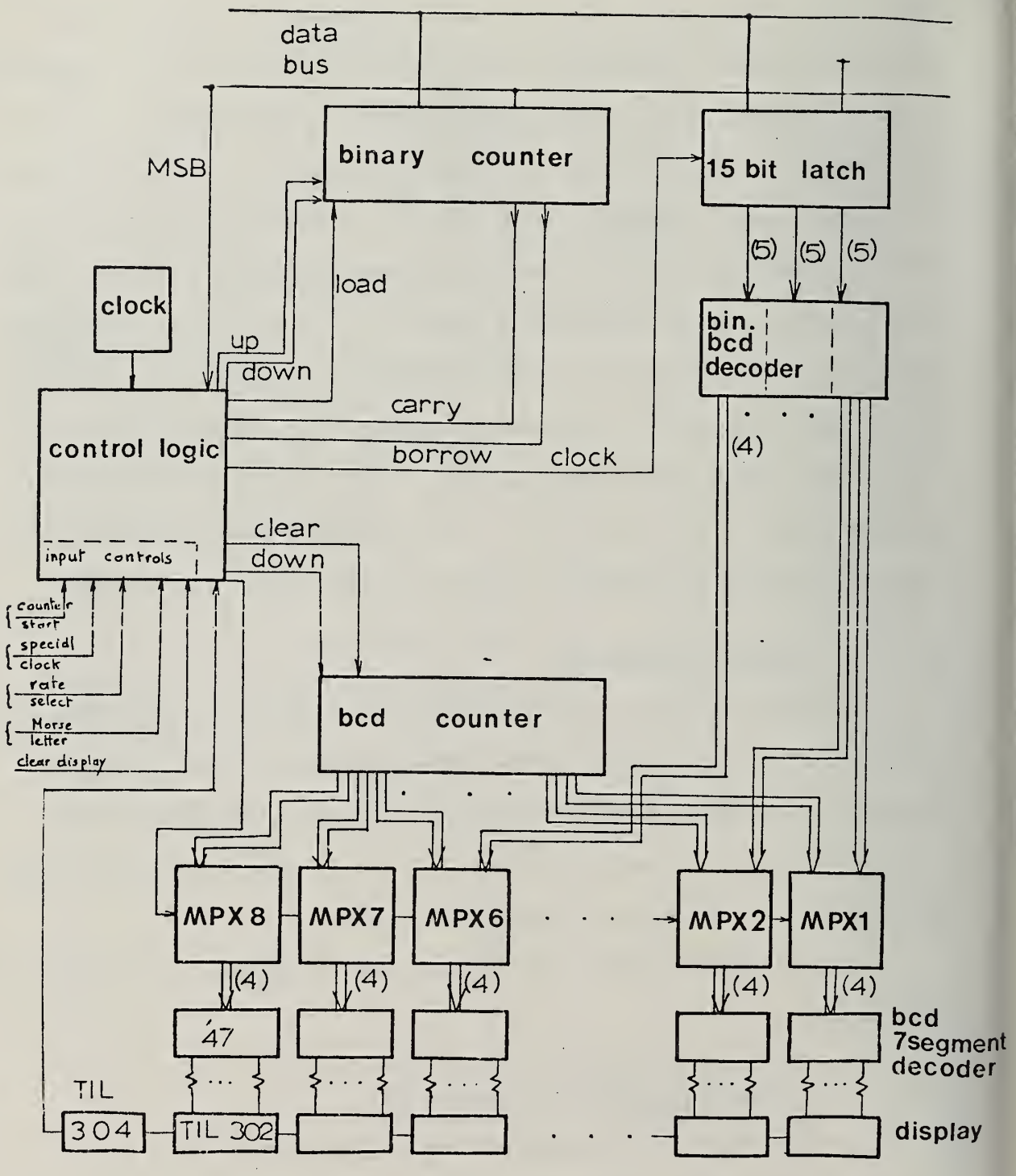


Figure 19. Block Diagram of the Display.

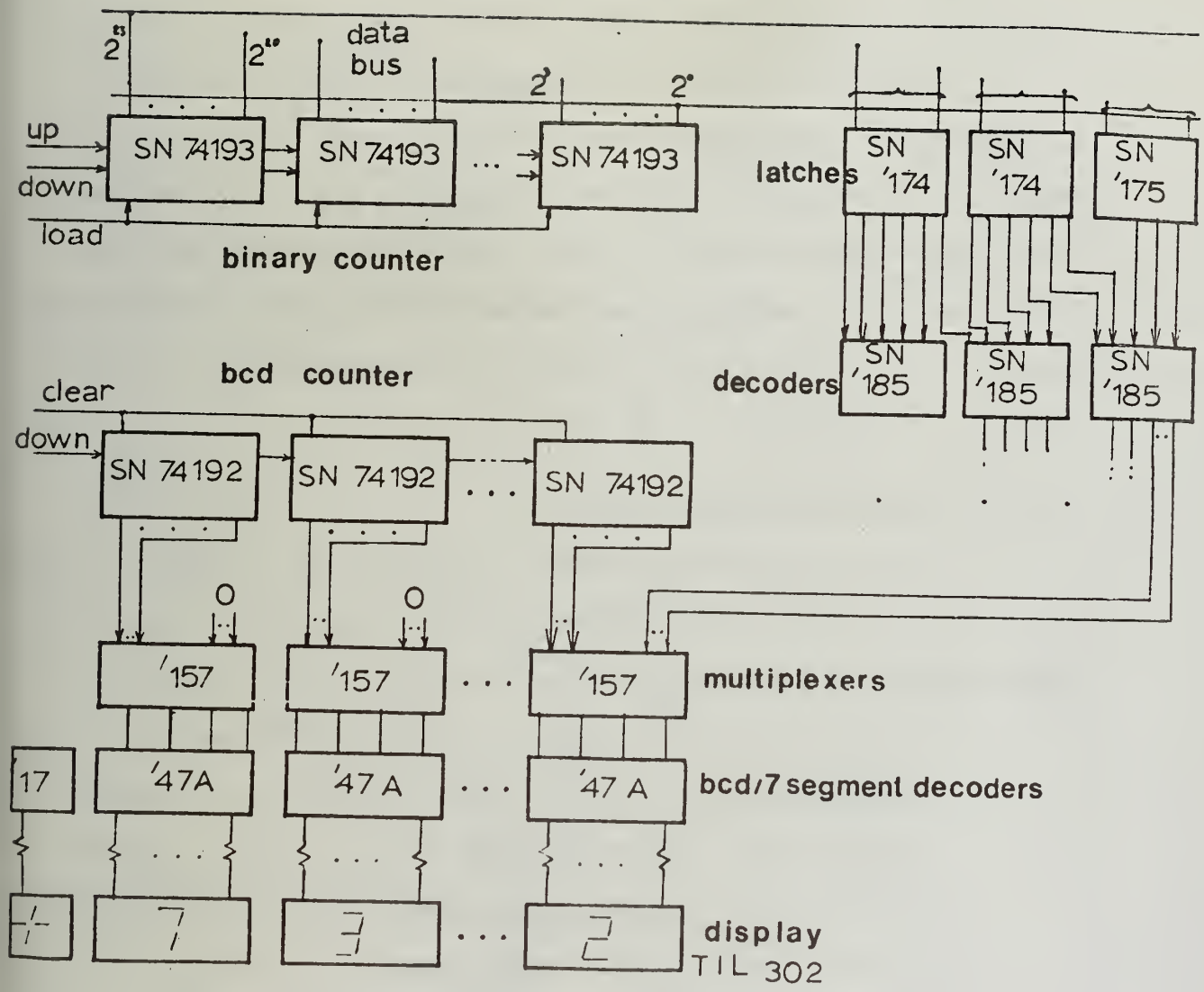


Figure 20. Conversion Circuits.

The circuits used for all the other conversions consists of a set of binary counters (SN 74 193) and a set of BCD counters (74 192).

Quadruple 2 line to 1 line multiplexers (SN 74 157) choose either the data coming from the BCD counters, or that coming from the binary to BCD ROMs. The multiplexers outputs are fed into BCD to 7 segments decoders and drivers (SN 74 47A). The decoder outputs drive the final stage, the 7 segment numeric display (type TIL 302). An additional circuit, the TIL 304 is used in order to indicate the sign. It uses a buffer driver.

3.2.2. The Control Logic (Figure 21)

When data are to be displayed, the instruction (a pulse on a given line) is decoded in order to perform particular functions which are of two types depending on whether the counters or the ROMs are to be used.

If we use the counters, three control lines are needed:

- counter start: this pulse loads the data into the binary counter, while clearing the BCD counters. Then it enables either the up or down line for the binary counter (for the BCD counter, only the down line is used).

  The carry and borrow lines of the binary counter are used to determine when it is empty. When this is detected, a pulse is produced which clears the flag and therefore disables the counting.

- special clock: this line is used when the data to be displayed requires a special conversion for angles or conversion of distance to feet). The flag is reset to zero when the binary counter is empty.

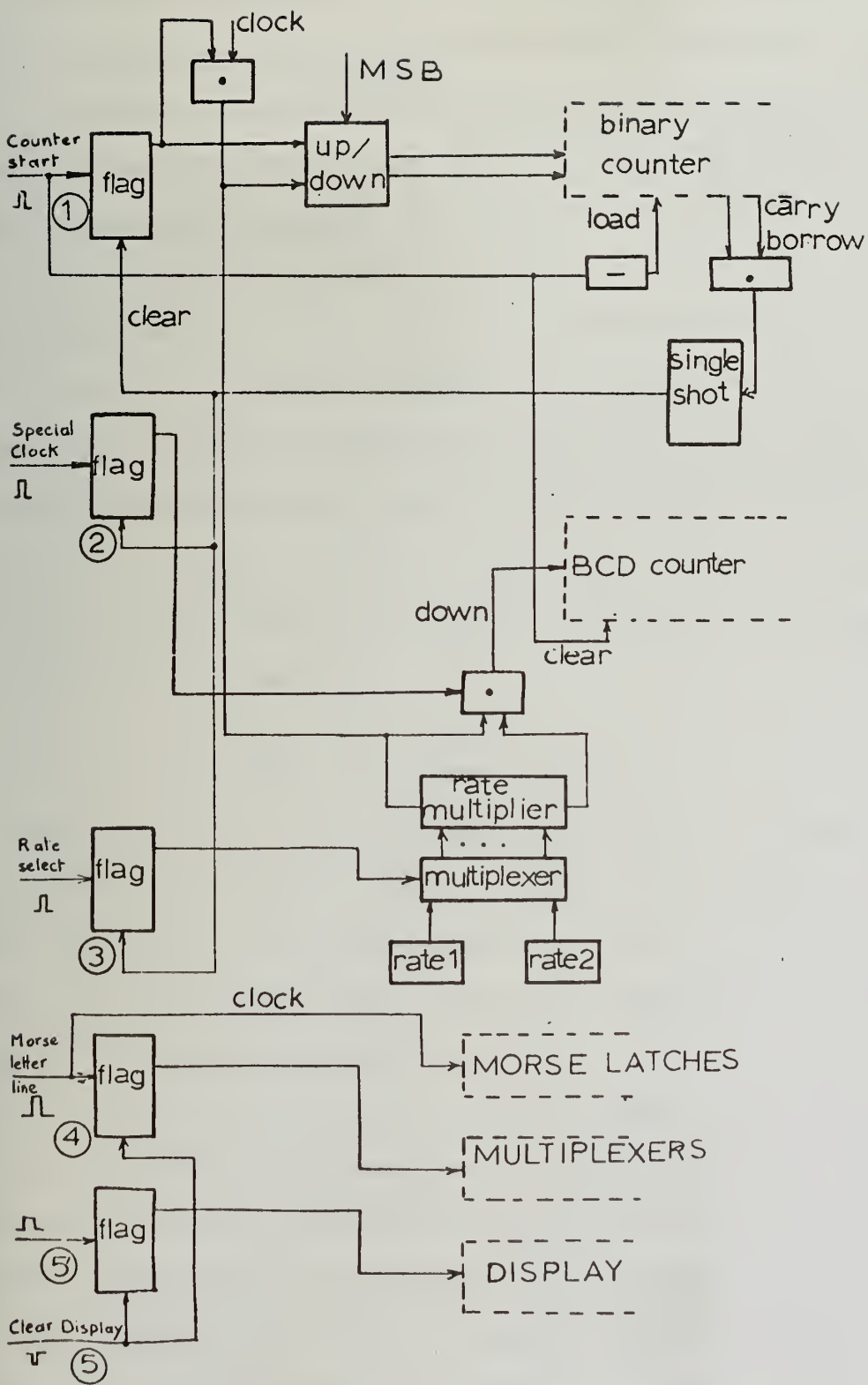


Figure 21. Control of the Display.

- rate select line: this line determines which rate must be set into the rate multiplier. The selection between rate 1 and rate 2 is done through 2 to 1 line data selectors. The flag is cleared when the binary counter is empty.

If the Morse code letter circuits are used, one line is required:

- Morse letter line: this line feeds the clock input of the latches and sets a flag which maintains the select line of the multiplexers which transmit either the outputs from the BCD counters or the outputs of the ROMs at 1.

A fifth line is used for turning off the display by clearing a flag whose output is fed into the Bl/RBO line of the 7 segment display circuits.

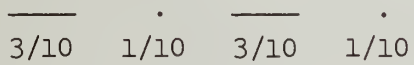
4. THE CODE KEYER

4.1. Definition of the Problem

Each radio station has an identifying code consisting of three letters (for Champaign it is CMI) which is transmitted using the Morse code. Using the same example, CMI is coded:

C M I

For the Morse code, the unit time used will be 1/10 second. The dash is coded 3/10 of a second and the dot as 1/10 of second. Within one letter, there is a blank of 1/10 of a second after each dash or dot. A blank separates two consecutive letters and lasts 3/10 of a second. Finally, between each complete station code, we have to provide a blank of approximatively 4 seconds. Example:



3/10 1/10 3/10 1/10

In addition, we must take into account that this three letter Morse code presents an additional letter, I, which is the same for all stations and which preceeds the 3 letter code (ICMI). The letter I appears when the pilot is approaching the runway. The code is generated by a narrow beam antenna which covers the runway.

The problem, then, is to generate an audio signal corresponding to the Morse code for a given station when the pilot tunes to its frequency. Since several stations use the same frequency (there are 200 of them available for radio stations at the present time), the CORDIC unit must determine which address in the PROMs corresponds to the most probable station which is emitting the signal we are receiving. Having done this, it will generate a signal indicating that we must load the code from the PROMs.

A major consideration for the Code Keyer is the choice of the code used to represent the Morse code. We know that the longest letter is Q which requires 13/10 of a second.

4.2. First Solution

A possible way of coding is the following:

00	for	.
01	for	—
10	for	end of a letter
11	for	end of the whole code

In addition to this, if after a dot or dash we find either 00 or 01 we automatically create a 1/10 of a second blank.

Now, we have to look for the longest possible word which contains 4 letters, the first of which is I and the others with a maximum number of 4 dashes or dots:

	bits
1st letter I	2
blank	2
2nd letter	2 x 4
blank	2
3rd letter	2 x 4
blank	2
4th letter	2 x 4
end of code	2

Therefore, we need 10 + 24 = 34 bits or 5 PROMs. (4 1/2). The system would appear as indicated in the Figure 22 for the first solution. This coding seems to be the one which requires the least amount of memory if we want to load the complete code in only one time, without using any other kind of memory such as a ROM.

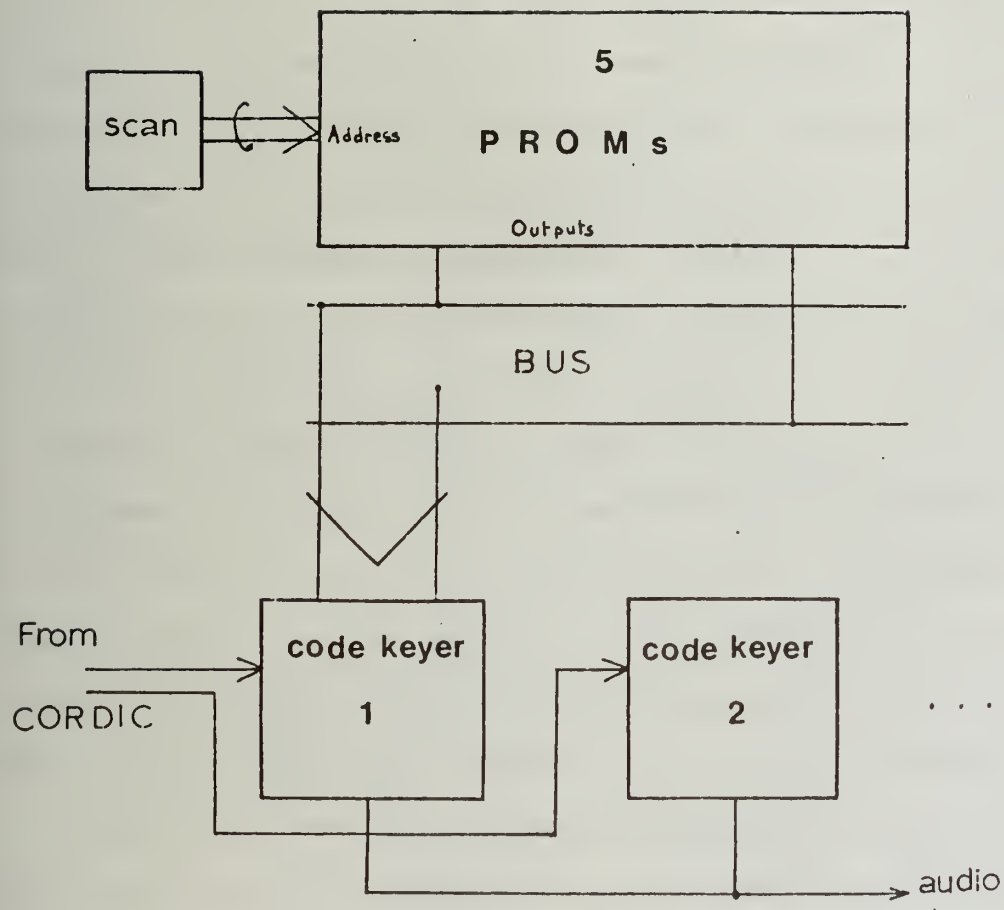


Figure 22. First Solution.

The outputs of the PROMs are connected to an 8 x 4 1/2 bus. For each desired radio, we use the same system (noted "code keyer radio i") which is enabled by the Cordic unit. The output of each of them delivers the audio signal in an unique loud speaker. Figure 23 presents a more detailed description of each code keyer. When a radio frequency is selected and Cordic has determined the address at which the corresponding data is located, a "data enable" signal is generated for this radio. This signal indicates that the right data is available at the outputs of the PROMs, and therefore on the bus. A enable device loads the information into a 34 bit shift register and disables the inputs after loading. Then, a decoder analyzes the first two bits of the shift register and generates either a 3/10 s or a 1/10 s 5V signal. Next, the decoder acts on the clock inhibit which allows two clock pulses to the shift register, therefore shifting right twice. The decoder analyzes these new right most bits, generating the corresponding signal. If these two bits represent either a dot or a dash, the device automatically generates a one unit time blank. When the code indicating the end of the whole Morse code occurs, the enable device is activated with a delay of 4 seconds in order to separate the repetitive Morse code. The output of the decoder is fed into an oscillator, then into an amplifier which drives a loud speaker.

This solution has the advantage of loading the entire Morse code for a given station which helps simplify the amount of control needed. But it has the disadvantage of needing 4 and a half PROMs which are still quite expensive (about 60 dollars each).

4.3. Second Solution

A cheaper way, which uses more logic is now presented. This solution requires the smallest possible number of PROMs. Each letter of the

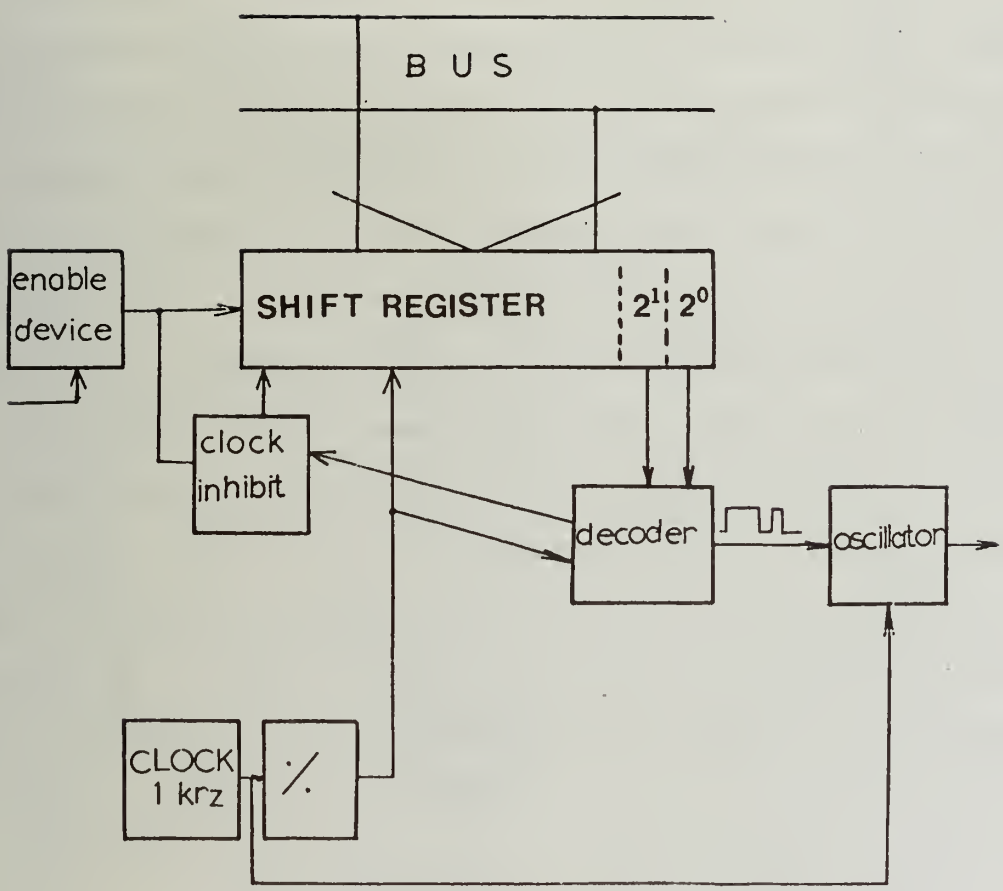


Figure 23. Block Diagram of the Code Keyer (Solution 1).

code is assigned a binary representation. Since there are 26 possible letters, the required number of bits is 5. Then we need 1 + 3 x 5 bits = 16 bits which requires two PROMs. In addition to this we need a Morse decoder. Such a system is shown in Figure 24. This system has several drawbacks. One of them is the fact that it is not possible to load the entire code in one operation from the two PROMs. This is due to the fact that the length of the coding of a Morse letter is variable. Thus we have to take the largest size (13 bits), which leaves unacceptable blanks. To avoid this we load only one letter at a time, send it to the code keyer and, when the corresponding audio signal has been generated, a flag enables the load of the next letter. The time to load it is at most the cycle time to scan the PROMs. This is on the order of one microsecond, which is negligible compared to a 1/10 of a second. The system works as follows: the enable signal indicates that data is available at the outputs of the PROMs. The letter select enables only one letter (5 bits). This letter addresses a Read Only Memory which contains the Morse alphabet whose coding is:

111	for	dash
1	for	dot
0	for	blank

For example, C would be coded as: 111010111101.

The resulting output is loaded into the right shift register, which is shifted right every 1/10 of a second, therefore producing a 3/10 of a second signal when three consecutive ones are found or a 1/10 of a second blank when zero is found. When the shift register is emptied of its useful information, we select the next letter which is loaded when data is determined to be present. This explains the operation of the system. In the following pages the various functions will be presented in more detail.

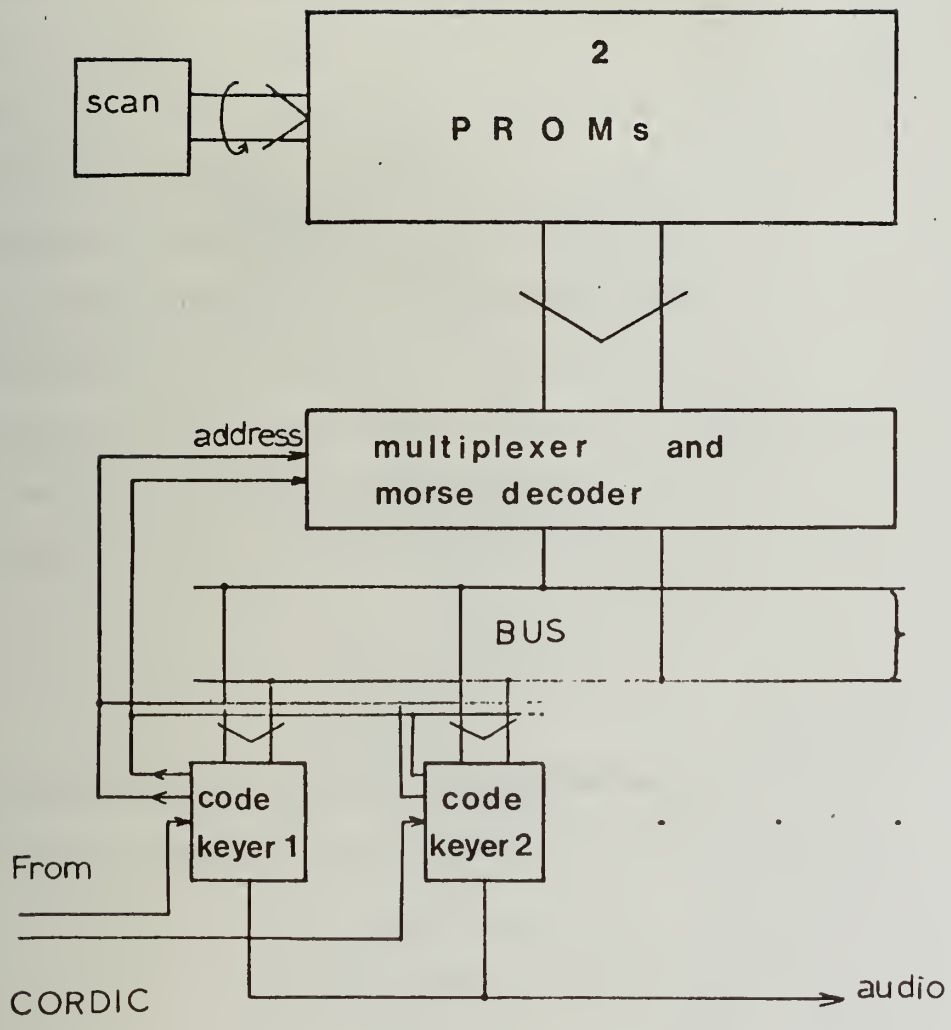


Figure 24. Block Diagram of Solution 2.

4.4. Multiplexing and Morse Decoder

Figure 25 shows the block diagram of the hardware which is shared by all the radios. It contains a selector which enables only one five bit letter among four. In fact, the first letter is coded 1 if there is a I, 0 if there is not, requiring only one bit in the PROMs. This bit is connected to the first input of the five multiplexers. When the multiplexer address is 00, the outputs are either 11111 or 00000 depending on whether there is an I or not. For the other letters, each bit is connected to the same input of each of the five multiplexers. Their outputs address two ROMs. Since the maximum information required to load a letter using the code as indicated above is 13 bits and the number of letters being 26, the use of two 32 x 8 bit ROMs satisfies the requirements (INTEL IM 5600). The outputs are open collector type. See Figure 26.

4.5. Code Keyer Card

For each radio, we need one card of this type which is independent from the others and has its own clock. Figure 27 shows the various parts of the code keyer card:

- shift register, data enable, clock inhibit
- 10 hz clock, divider
- end of letter detector
- letter counter
- end of code detector
- multiplexer address selector
- jump if no I
- sequence device
- initial clear device

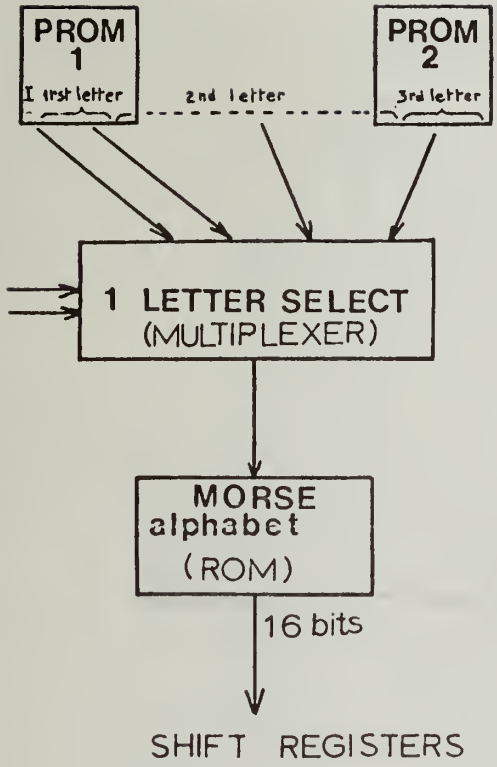


Figure 25. Block Diagram of the Letter Select and Morse Decoder.

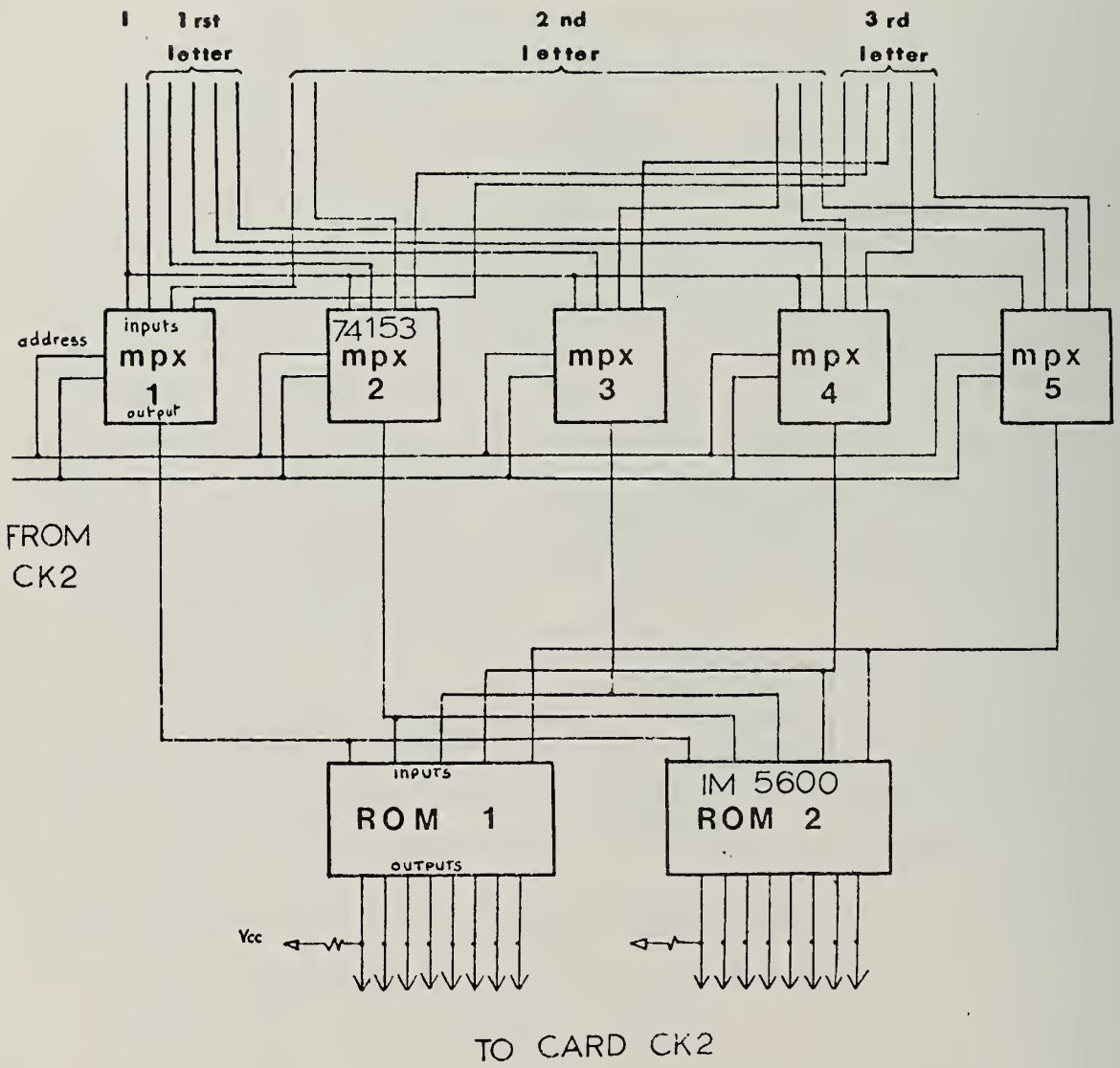


Figure 26. Card CK1: Implementation of the Letter Select and Morse Decoder.

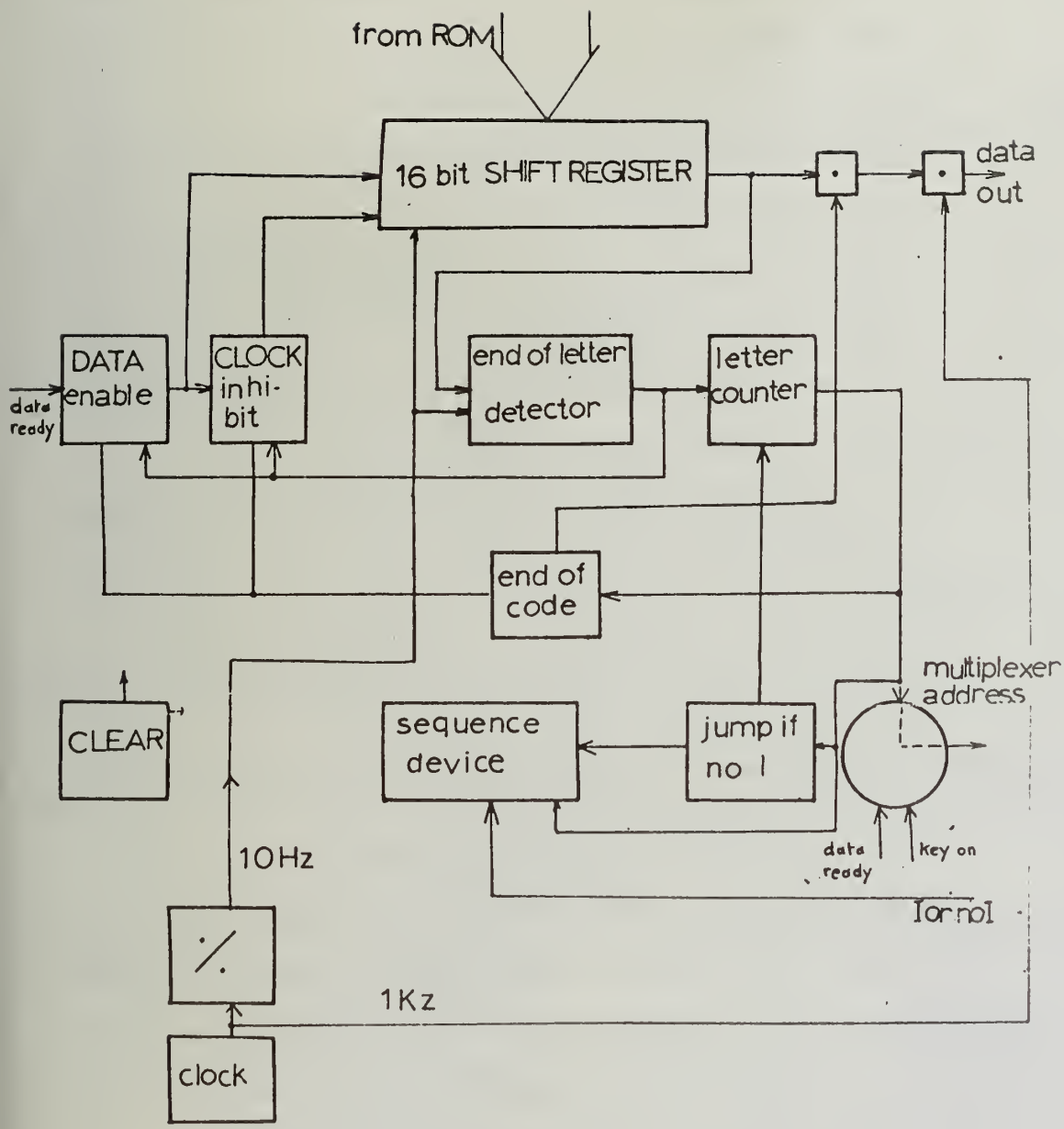


Figure 27. Block Diagram of the Code Keyer.

Before giving details concerning each of the above functions, the general processing of the data through them will be discussed. See Figure 28.

The input signals are:

- 16 data bits coming from the Morse ROMs.
- the first bit of the PROMs which indicates the presence or absence of the letter I.
- two control signals:
  - "data ready," which indicates that the data at the outputs of the PROMs are those corresponding to our radio station.
  - "key on," which indicates that a radio has been set at a given frequency and that the pilot desires to hear its corresponding Morse code.

The output signals are:

- "data out," which is the output of the shift register, modulated with a 1 khz digital clock.
- the two line address of the multiplexers, for selecting the next letter to be loaded in the shift register.

Suppose that the pilot tunes a radio. The signal "key on" goes to zero. This makes the Clear Device clear the general flipflops contained in the system. When the Data Ready pulse arrives, it causes the Data Enable circuit to load the data coming from the ROMs into the shift registers. When assured of the loading, the Clock Inhibit enables the clock to the shift registers at a frequency of 10 Hz. The right most bit of the shift register is fed into a device clocked at the same frequency which generates a signal when all the code has been shifted out for a given letter (this circuit is called the "end of letter detector"). Each of these pulses increments a binary counter which counts the number of letters which have been processed. Its outputs are also used for other purposes. The first is to

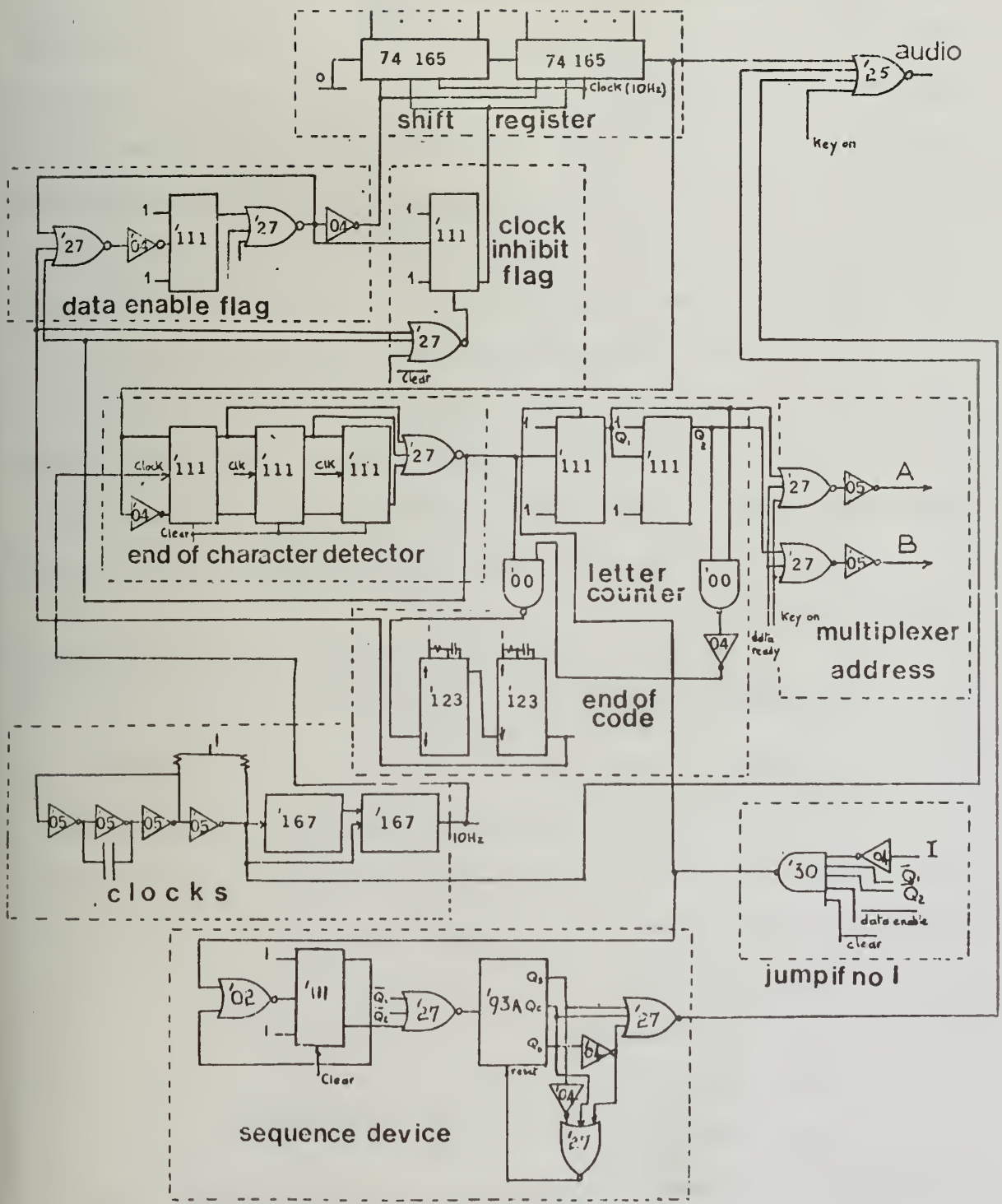


Figure 28. Circuit Diagram of the Code Keyer.

address the one letter select multiplexers. Another function is to inform the "jump if no I" circuit when the next letter to be loaded is 00 or the first bit of the PROMs. If this bit is zero, meaning that there is no I, the letter counter will be set at 01 in order to be ready to address the next letter (first letter in this case since there is no I). One pecularity of the repetition of the identification Morse code for a station is the following. If there is an I, we generate the code in a continuous manner. If there is no I, we send the code four times followed by a blank whose time length is the same as the code itself. All these required sequences are determined by the "sequence device" which uses the data from the first bit of the PROMs, the "jump if no I" circuit and the outputs of the letter counter. In addition to this, a circuit (end of code) disables the loading during 4 seconds, after a complete code has been generated.

In the following, the different devices which constitute the Code Keyer are discussed.

4.6. Code Keyer Circuits

4.6.1. Clear Device (Figure 29)

When the radio has been tuned, the signal "key on" is set at zero. At this time a narrow pulse is generated which resets the flipflop devices. The element cleared at this time are:

- letter counter
- end of code detector
- clock inhibit flag
- data enable flag

The circuit used is an SN 74 123 monostable multivibrator with clear, an external resistor, a diode and a capacitor.

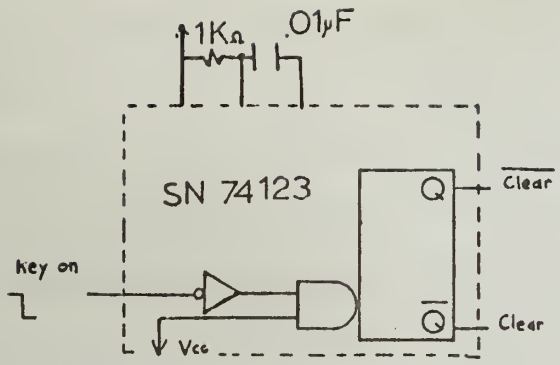


Figure 29 . Clear Device.

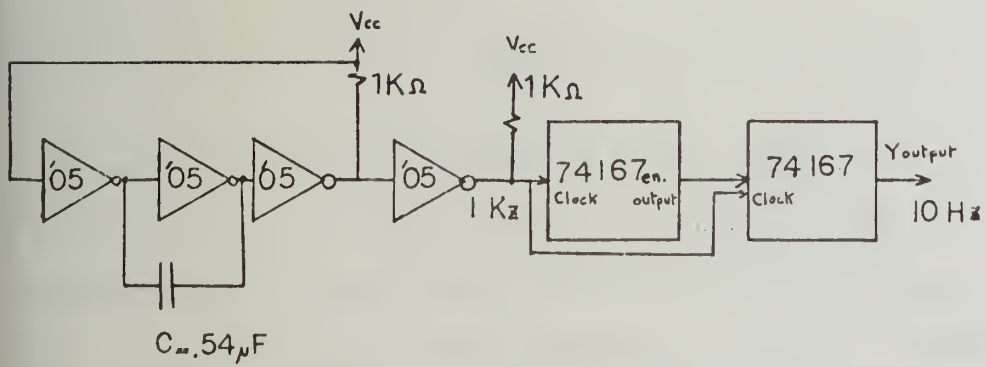


Figure 30. 1 Khz and 10 Hz Clocks.

4.6.2. The 1 kHz and 10 Hz Clocks (Figure 30)

Each code keyer card is equipped with its own clock. This must be a low frequency, 10 Hz . Since we need an audio signal of 1 kHz to modulate the signal coming out of the shift register, we design a 1 kHz clock and obtain the 10 Hz by frequency division. The circuit used to generate the 1 kHz is the one indicated on Figure 30. It uses three cascaded open collector inverters connected together to form a loop. The signal obtained is such that the off to on time ratio is 5/6 and the frequency is given by:

$$f = \frac{1}{t_1+t_2} = \frac{1}{t_1(1+6/5)}$$

$$t_1 = CR_4 \ln\left(\frac{V_{CC} - V_{BE} - V_{CEsat}}{V_{CC} - V_{BE} - V_t}\right)$$

$$t_2 = CR_1 \ln\left(\frac{V_t - V_{CEsat}}{V_{CC} - V_{BE} - V_t}\right)$$

This circuit has the drawback of being dependent on the source voltage V_{CC}. For a capacitor C = .5μF we obtain the desired frequency. In order to improve the shape of the output signal, an additional open collector inverter is used. This yields very short transition times.

To obtain the 10 Hz clock, we divide this frequency two consecutive times by 10. The circuits used are two SN 74 167 synchronous and programmable counters which are easily cascaded by connecting the enable output to the enable and strobe inputs of the next stage.

4.6.3. Shift Register - Data Enable Flag - Clock Inhibit Flag (Figure 31)

The data coming from the Morse ROMs are connected to the parallel inputs of two serial shift registers. Loading is controlled by the load/shift input of the shift register.

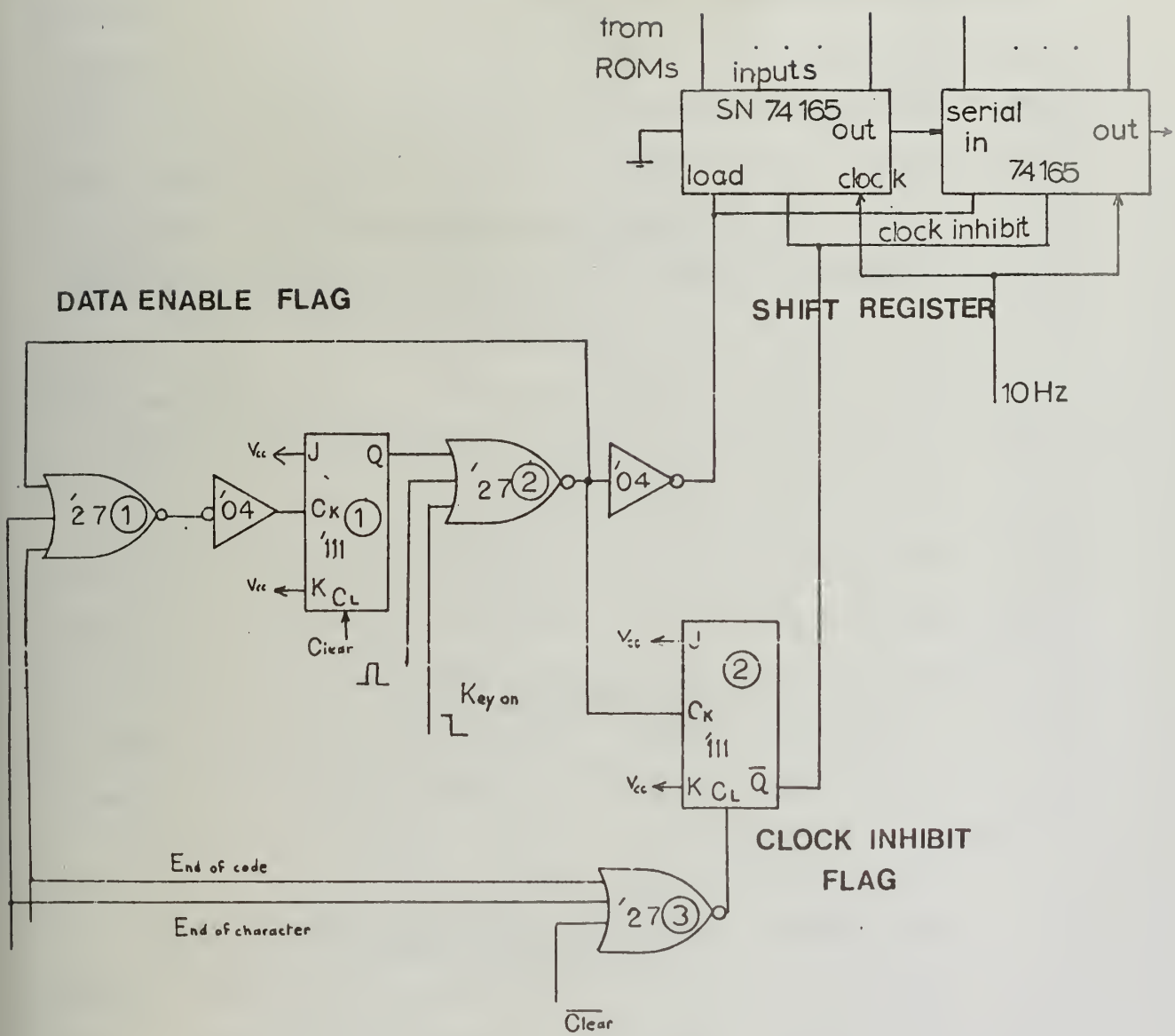


Figure 31. Shift Register-Data Enable and Clock Inhibit Flags.

When a radio is tuned, the clear device sends a negative pulse which clears flipflop 1 making the Q output 0. When "data ready" is present (meaning that the code corresponding to our radio is available at the outputs of the PROMs), a positive pulse, whose length is the same as "data ready," appears at the output of the three input NOR gate 2. This pulse, through an inverter, holds the load/shift line of the shift registers to zero for a period equal to the length of the pulse, causing the data at the parallel inputs to be loaded in. The falling edge of this same positive pulse, sets the output of flipflip 1 to 1, disabling the load line prior to the next "data ready pulse." At the same time (falling edge), the output of flipflop 2 changes state and enables the shift register clock, making it start. Flipflop 2 is initially cleared, disabling the shift register clock. It is also cleared when the end of character signal is one, indicating that the data in the shift registers has been unloaded and that shifting should stop to prepare for another load. Similarly, the loading is enabled by the falling edge of the end of character signal or by the end of code signal which occurs after an entire Morse code has been generated.

4.6.4. End of Character, End of Code Detectors - Letter Counter

The "end of character detector" detects when the end of the data contained in the shift register occurs. Since the data is either 111 or 1 or a single 0, end of data occurs when two consecutive zeros follow a one. Figure 32 shows this logic. It consists of three cascaded flipflops whose common clock is the 10 Hz one. The input is the serial output of the shift register. In order to determine the sequence 100 we enter the Q input of the first two flipflops and the complemented Q of the third one into a NOR gate which generates a positive pulse when the desired sequence occurs.

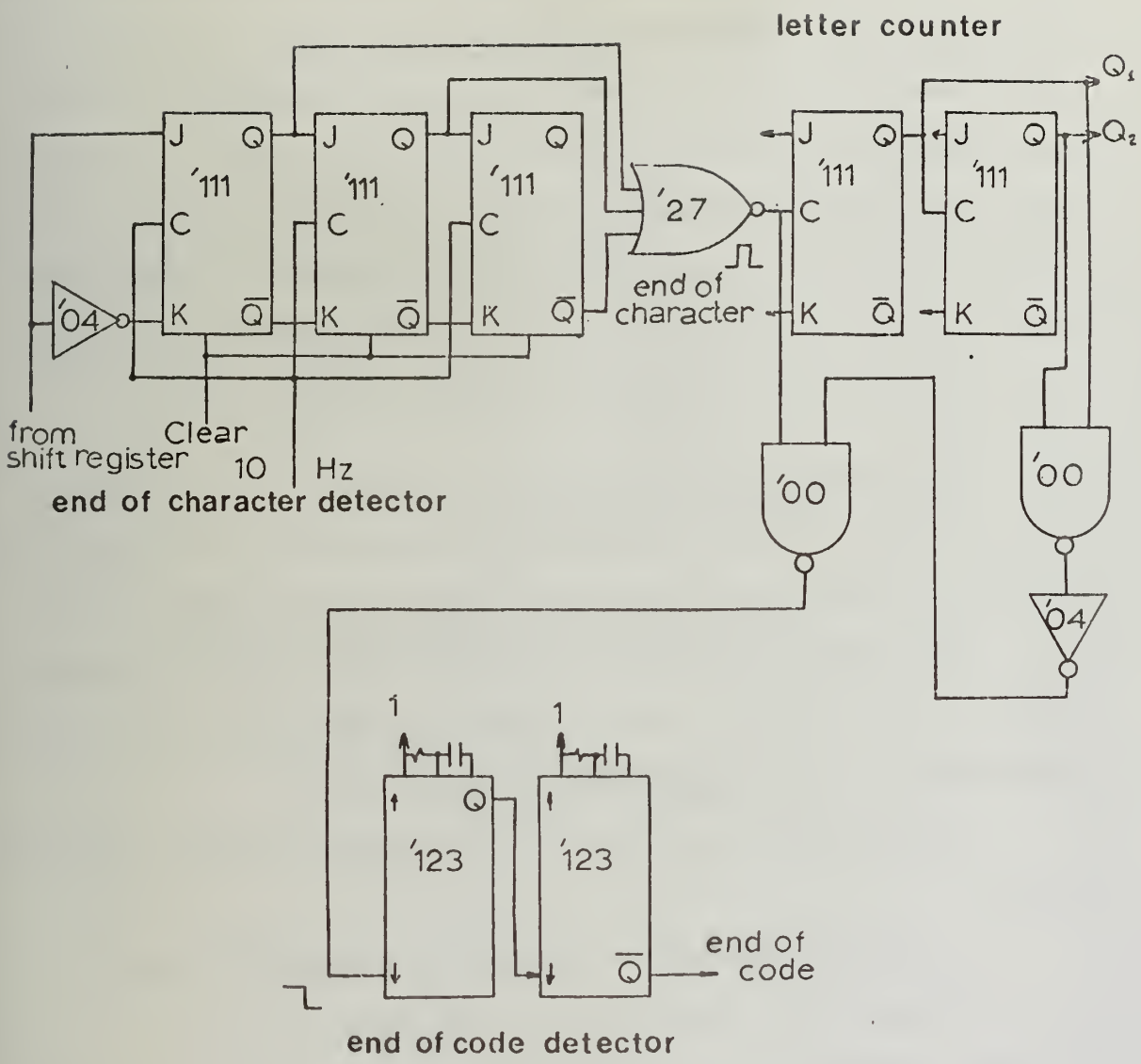


Figure 32. End of Character Detector-End of Code Detector-Letter Counter.

The falling edge of this pulse triggers a two bit counter which memorizes the next letter taken from the Morse code. When the outputs Q1 and Q2 are one (indicating that the letter on the way is the last one) and when the end of character signal is one (meaning the code is completed) a falling edge triggers a 20 ns single shot (monostable multivibrator). The falling edge of this short pulse triggers another single shot which has a pulse length of 4 seconds which is the blank required between two consecutive codes. This pulse, called the "end of code," disables the "data enable" circuit for 4 seconds, as well as the clock inhibit.

4.6.5. Multiplexer Address Selector (Figure 33)

As indicated by its name, this circuit provides the selection of the address of the desired letter to be loaded into the shift registers.

This address consists of two bits which are 1 when one or more radios are not set. The selector uses open collector components so that the address select for all radios are wire ORed.

The following signals are required to load the letter address onto the bus:

- a "data ready" pulse corresponding to the desired radio,
- "key on" set, indicating that this radio is used.

The result, when these two conditions are met, is to transfer to card CK1 the values which are at the outputs of the letter counter. The two inverters used to complement the outputs of the NOR gates are open collector type. If either "data ready" or "key on" or both are 1, the outputs are 1.

4.6.6. Jump if No I (Figure 34)

When the letter counter is at 00, we are ready to load the first letter of the Morse code (I or no I). When the signals "data ready" and "key on" are set, we check the first letter:

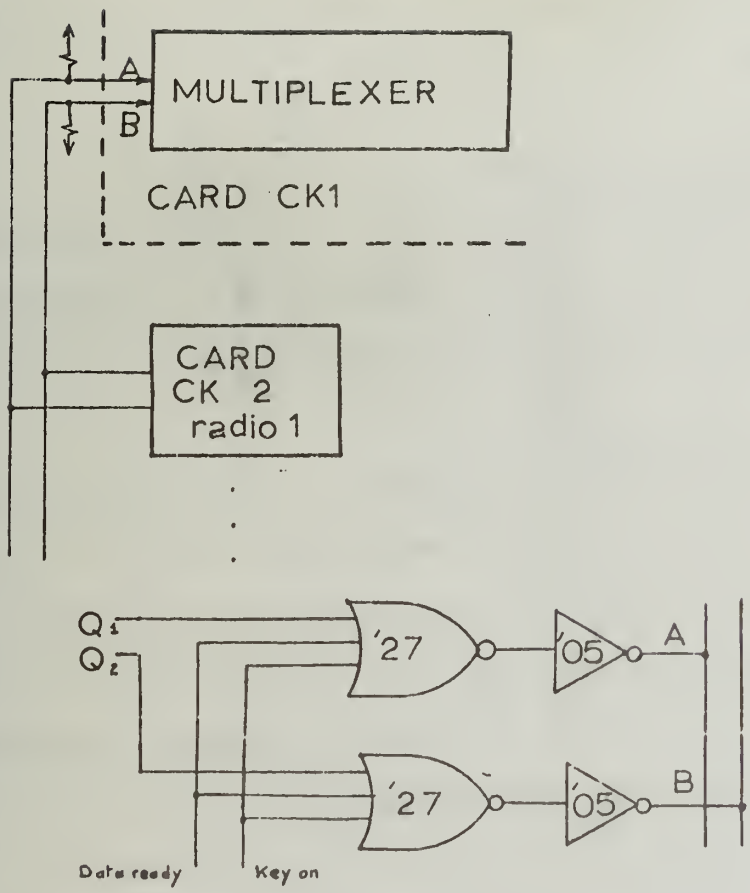


Figure 33. Multiplexer Address Selector.

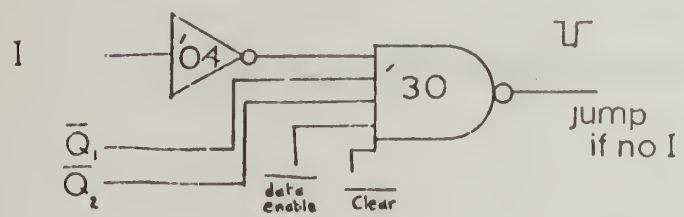


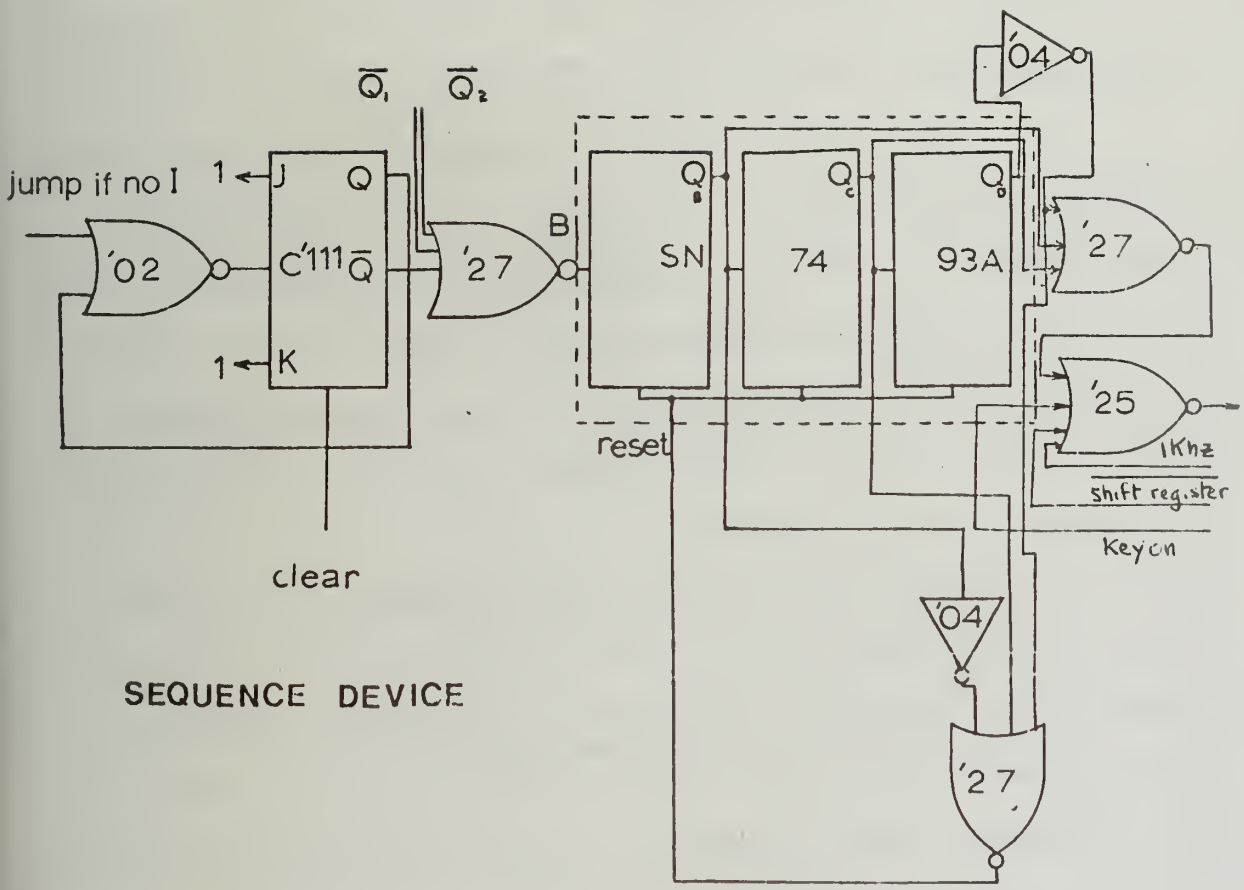
Figure 34. Jump if no I.

- if it is 0 (no I) we set the letter counter to 01 in order to load the next letter (therefore, only Q1 is changed).
- if it is 1 (there is an I), no particular action is done and we load it as any other letter.

4.6.7. Sequence Device (Figure 35)

The purpose of this circuit is to transmit the Morse code in a continuous manner if there is an I. If there is no I, the code is transmitted four times followed by a blank which lasts the same time as the code itself.

In order to realize this function, we use the output of the logic noted "jump if no I" which generates a negative pulse when no I is found. The circuit is shown in Figure 35. The flipflop shown is initially cleared. Its Q output is fed back to the clock input together with the jump if no I pulse. When this pulse arrives, its rising edge changes the output of flipflop 1 and, due to the feedback, this output will stay at one as long as we do not change the frequency of the radio. When the outputs of the letter counter go to 00, the NOR gate output changes to 1. When Q1 goes to 1, the output of the NOR gate falls to 0 and this transition increments an SN 7493 three bit counter. When its Q_B Q_C Q_D outputs are 100 (meaning that the Morse code has been sent four times), a NOR gate which has these signals as inputs disables the output of the shift register and a blank is generated. After detection of the 100 output, this counter is reset to 000.



SEQUENCE DEVICE

Figure 35. Sequence Device.

5. CONCLUSION

Some operations of the Radio Aids Simulator have been discussed in the preceding pages. A prototype Code Keyer system was fabricated and proved operational.

The implementation of the system presents several unique features. The main one is the large use of ROMs, not only for storing data, but also for reducing the amount of combinatorial logic in the control systems. This is made possible by the constant decrease in the price of integrated circuits, while they are simultaneously becoming faster and smaller. In addition, each year new integrated circuits become available which implement even more complex functions.

For each function of the Radio Aids discussed in this paper, several solutions using digital circuits have been presented. This establishes that the Radio Aids can be implemented in digital form using a special purpose computer, and provides a partial solution to one of the major problems in flight simulation, the need for large digital computers.

LIST OF REFERENCES

1. Texas Instruments Incorporated, "The TTL Data Book for Design Engineers," by the Engineering Staff of T. I.

2. Kohavi, "Switching and Finite Automata Theory," McGraw Hill Computer Science Series, 1970.

3. Poppelbaum, W. J., "Computer Hardware Theory," Macmillan, 1972.

4. Institute of Aviation - University of Illinois, "Fundamentals of Aviation and Space Technology," Institute of Aviation of the University of Illinois, 1971.

5. D. A. Hodges, "Semiconductors Memories," IEEE Computer Society, 1972.

6. Millman and Taub, "Pulse, Digital and Switching Waveforms",McGraw Hill, 1965.

IOGRAPHIC DATA T	1. Report No. UIUCDCS-R-74-633	2.	3. Recipient's Accession No.
le and Subtitle DIGITAL RADIO AIDS FOR A FLIGHT SIMULATOR RADIO MEMORIES - DISPLAY - CODE KEYER			5. Report Date May 1974 6.
hor(s) Lucien Isidore Facchin			8. Performing Organization Rept. No. UIUCDCS-R-74-633
forming Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
nsoring Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			13. Type of Report & Period Covered technical 14.

pplementary Notes

stracts

The research described here is part of a joint study between the Department of puter Science and the Institute of Aviation of the University of Illinois. This ly is a result of the need to consider alternatives to using general purpose digital puters for the solution of the entire aircraft simulation problem. The software s of such systems have sky-rocketed in recent years while hardware costs continue all. Thus, serious consideration must be given to direct functional mechanization ough the use of special purpose computers. The use of small dedicated computers may tly simplify interfacing to the central control computer in addition to reducing amount of software required by difusing the processing throughout the simulator. s document presents an implementation of some of the functions of such a system, the io Aids System.

y Words and Document Analysis. 17a. Descriptors

tal radio
ght simulation
io aids

dentifiers/Open-Ended Terms

GSA II Field/Group

ailability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 72
nlimited distribution	20. Security Class (This Page UNCLASSIFIED	22. Price

NTIS-35 (10-70)

USCOMM-DC 40329-P71